人工智能与机器学习

hanbing Qin

2026 年 06 月

目录

1 机器学习的定义、分类和方法要素

机器学习：是关于计算机基于数据 构建概率统计模型 并运用模型对数据进行预测与分析的一门学科。

- 研究对象是数据，从数据出发，提取数据的特征，抽象出数据的模型，发现数据中的知识，又回到对数据的分析与预测中去。
- 基本假设：同类数据具有一定的统计规律性。
- 方法：1、给定的、有限的、用于学习的训练数据出发；

  2、模型——属于某个函数的集合，称为假设空间（hypothesis space）；

  3、从假设空间中选取一个最优模型，使它对已知的训练数据及未知的测试数据（test data）在给定的评价准则下有最优的预测

1.1 监督学习（Supervised Learning）

定义：从**标注数据**中学习预测模型的机器学习问题

- 标注数据表示输入输出的对应关系
- 预测模型对给定的输入产生相应的输出
- 输入到输出的映射的统计规律
- 标注的数据集往往是人工给出的，所以叫监督学习
- 利用训练数据集学习一个模型，再用模型对测试样本集进行预测，因此分为学习系统和预测系统。

数据：

- 输入/特征：随机变量 X；每个具体的输入是一个实例（instance），取值为特征向量 x
- 输出：随机变量 Y；每个具体的输出是一个标签（label），取值为特征向量 y
- 样本点：输入-输出对 $(x_i\ y_i)$
- 训练数据：学习模型使用的输入-输出集合
- 测试数据：模型需要预测的输入-输出集合
- 独立同分布假设

1、输入变量与输出变量均为连续变量的预测问题称为**回归问题**
2、输出变量为有限个离散变量的预测问题称为**分类问题**
3、输入变量与输出变量均为变量序列的预测问题称为**标注问题**

模型：

- 输入到输出的映射，这一映射由来表示

- 假设空间：模型属于由输入空间到输出空间的映射的集合；假设空间的确定意味着学习的范围的确定
- **模型类别**：概率模型或非概率模型
- 概率模型：条件概率分布 $P(Y|X)$
- 非概率模型表示：决策函数 $Y=f(X)$
- 对具体的输入进行相应的输出预测时，写作 $P(y|x)$ 或 $y=f(x)$
- 一般用参数化的模型 $P(Y|X;\theta)$ 和 $Y=f(X;\theta)$

策略：

- 按照什么样的准则学习或选择最优的模型
- **损失函数**: 度量模型一**次**预测的好坏
- **期望损失**：损失函数的期望（理论上模型关于联合分布 $P(X,Y)$ 的平均意义下的损失），但由于联合分布未知，因此无法计算……

$$R_{exp}(f)=E_P[L(Y,f(X))]=\int_{x*y}L(y,f(x))P(x,y)dxdy$$

- **经验损失**：损失函数在训练集上的平均（模型 $f(x)$ 关于训练数据集的平均损失）

$$R_{emp}(f)=\frac{1}{N}\sum_{i=1}^{N}L(y_i,f(x_i))$$

- **当样本容量 N 趋于无穷时，$R_{emp}(f)$ 趋于 $R_{exp}(f)$，用经验损失估计期望损失**因此：
- **经验风险最小化**：认为经验损失最小的模型是最优的模型，即求解最优化的问题：

$$\min_{f\in F}\frac{1}{N}\sum_{i=1}^{N}L(y_i,f(x_i))$$

  注意：

  当样本容量足够大时，经验损失最小化能保证有很好的学习效果，在现实中被广泛采用

  当样本容量很小时，经验损失最小化未必好，会产生“过拟合”（over-fitting）现象

- **结构风险最小化**：是为了防止过拟合而提出来的策略，加入用正则项（regularization）近似期望损失函数；$J(f)$ 为模型的复杂度，权衡经验风险和模型复杂度，希望同时小。
- 求解最优化问题：

$$\min_{f\in F}\frac{1}{N}\sum_{i=1}^{N}L(y_i,f(x_i))+\lambda J(f)$$

评估：

- 目的：学到的模型不仅对已知数据而且对未知数据都能有很好的预测能力。
- **训练误差**：训练好的模型 $Y=f(X;\hat{\theta})$ 关于训练数据集的平均损失

$$R_{emp}(\hat{f})=\frac{1}{N}\sum_{i=1}^{N}L(y_i,\hat{f}(x_i))$$

- **测试误差**：是模型 $Y=f(X;\hat{\theta})$ 关于测试数据集的平均损失

$$e_{test}=\frac{1}{N'}\sum_{i=1}^{N'}L(y_i,\hat{f}(x_i))$$

1、训练误差的大小对判断给定的问题是不是一个容易学习的问题是有意义的，但本质上不重要

2、测试误差反映了学习方法对未知的测试数据集的预测能力，是学习中的重要概念。希望测试误差小

3、通常将学习方法对未知数据的预测能力称为**泛化能力**

选择：

- 过拟合：一味追求提高对训练数据的预测能力，所选模型复杂度则往往会比真模型更高
- 欠拟合：一味追求降低模型复杂度，会损失对训练数据以及测试数据的预测能力
- 模型选择旨在**避免过拟合和欠拟合**

1.2 无监督学习（Unsupervised Learning）

整体思路与监督学习类似，就不详细写了……

定义：是指从无标注数据中学习预测模型的机器学习问题

- 无标注数据是自然得到的数据
- 预测模型表示数据的类别、转换或概率
- 本质是学习学习数据中的统计规律或潜在结构
- 模型可以实现对数据的聚类、降维或概率估计

数据：相对于监督学习而言，输出变成了隐藏随机变量 Z，组成希望找到的隐含结构空间，取值为隐含向量 z

模型：输入到隐含变量的映射；$P(Z|X)$、$P(X|Z)$ 或 $Z=f(X)$

2 线性回归

2.1 最小二乘线性回归

线性回归：假定输入输出变量间的函数关系 $y = f(x;\theta)$ 是线性的。**本质上是参数 θ 的线性函数**。

数据：

- 输入向量 $x = (x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(p)})^T$
- 输出标量（可也拓展为向量）y
- 数据集 $\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$

模型：

$$y = \theta_0 + \theta_1 x^{(1)} + \theta_2 x^{(2)} + \cdots + \theta_p x^{(p)}$$

$$y = \theta_0 + \sum_{j=1}^{p} \theta_j x^{(j)}$$

$$y = x^T \theta$$

- 增广的输入向量 $\boldsymbol{x} = (1, x^{(1)}, x^{(2)}, \dots, x^{(p)})^T$
- $\boldsymbol{\theta} = (\theta_0, \theta_1, \dots, \theta_p)^T$ 是待学习的参数向量，$p+1$ 行 1 列
- **模型是参数 θ 的线性函数**，确定了线性函数组成的假设空间

策略：

- 最常用的损失函数是平方损失 $L(y, f(x)) = (y - f(x))^2 = (y - x^T\theta)^2$
- 对应的经验风险 (训练集上的平均损失)：$T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$，　$R_{\text{emp}}(f) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i))$
- RSS（残差平方和）

$$RSS(\theta) = \sum_{i=1}^{N} (y_i - f(x_i))^2 = \sum_{i=1}^{N} (y_i - x_i^T\theta)^2$$

- 经验风险的设置并没有假设线性模型的有效性，仅仅是目标寻找数据的最佳线性拟合。
- 将所有数据表示成矩阵：$y = (y_1, y_2, \dots, y_N)^T$

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \dots & x_{Np} \end{pmatrix} = (x_1, x_2, \dots, x_N)^T$$

$\theta = (\theta_0, \theta_1, \theta_2, \ldots, \theta_p)^T$

则

$$RSS(\theta) = (\boldsymbol{y} - X\theta)^T(\boldsymbol{y} - X\theta)$$

算法：直接对 $RSS(\theta) = (\boldsymbol{y} - X\theta)^T(\boldsymbol{y} - X\theta)$ 式对 θ 求一阶导为 0 有，解得最小值成立 $\frac{\partial RSS}{\partial \theta} = -2X^T(\boldsymbol{y} - X\theta) = 0$ 为：

$$X^T(\boldsymbol{y} - X\theta) = 0 \Rightarrow X^TX\theta = X^T\boldsymbol{y} \Rightarrow \hat{\theta} = (X^TX)^{-1}X^Ty$$

解出 $\hat{\theta}$ 后即可对新的向量 x_{new} 进行预测：$\hat{y} = x_{new}^T\hat{\theta}$

- 对于式子：$X^TX\theta = X^T\boldsymbol{y}$ 是一个关于 θ 的线性方程组，若方程组的个数少于参数的数目，则 θ 有无穷个解；方程组的个数为样本点数目 N，参数的数目为 $p+1$
- $N < p+1$ **时** X^TX **不可逆，发生过拟合**，可通过减少输入的数目（特征选择）或者正则化来处理

模型：输入输出写为随机变量：$Y = X\theta + \epsilon$，其中噪声 ϵ 的均值为 0，方差为 σ^2，当噪声服从**正态分布**时，有 $\epsilon \sim N(0, \sigma^2)$，对噪声方差 σ^2 的估计为：

$$\hat{\sigma}^2 = \frac{1}{N-p-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

随机变量 $\hat{\theta}$ 满足正态分布：$\hat{\theta} \sim N\left(\theta, (X^TX)^{-1}\sigma^2\right)$，由此可以推导得到第 j 个 θ 值其近似的 95% 置信区间（置信区间包含 0 则不显著）：

$$\left(\hat{\theta}_j - 1.96v_j^{\frac{1}{2}}\hat{\sigma},\ \hat{\theta}_j + 1.96v_j^{\frac{1}{2}}\hat{\sigma}\right)$$

其中 $v_j^{\frac{1}{2}}\hat{\sigma}$ 是近似的标准误差。（概统里面的正态的置信区间即为 $\left[\bar{X} - z_{\alpha/2}\cdot\frac{\sigma}{\sqrt{n}},\ \bar{X} + z_{\alpha/2}\cdot\frac{\sigma}{\sqrt{n}}\right]$，这里的 $v_j^{\frac{1}{2}}\hat{\sigma}$ 可以理解成相当于标准差 $\frac{\sigma}{\sqrt{n}}$）

再进行 F 检验，这里可以先回顾一下概统里面的 F 统计量：

若 A、B 均服从 χ^2 分布且独立

$$A \sim \chi^2(df_1),\ B \sim \chi^2(df_2)$$

则

$$F = \frac{A/df_1}{B/df_2} \sim F(df_1, df_2)$$

此处要同时**检验多组系数的显著性**，即检验是否能将 $p_1 - p_0$ 个变量从一个模型中同时被排除:

由于 $\epsilon \sim N(0,\sigma^2)$，则残差平方和 $RSS=\sum_{i=1}^{N}(y_i-f(x_i))^2 \sim \sigma^2\chi^2(N)$

$$\frac{RSS_1}{\sigma^2} \sim \chi^2(N-p_1-1)$$

$$\frac{RSS_0-RSS_1}{\sigma^2} \sim \chi^2(p_1-p_0)$$

所以有 F 统计量:

$$F=\frac{\left(\frac{RSS_0-RSS_1}{\sigma^2}\right)\Big/(p_1-p_0)}{\left(\frac{RSS_1}{\sigma^2}\right)\Big/(N-p_1-1)}=\frac{(RSS_0-RSS_1)/(p_1-p_0)}{RSS_1/(N-p_1-1)} \sim F_{p_1-p_0,\ N-p_1-1}$$

- RSS_1 是含 p_1+1 个参数的大模型的残差平方和
- RSS_0 是选自 RSS_1 中含 p_0+1 个参数的较小模型的残差平方和
- F 统计量度量了在较大模型上每增加一个参数后, 残差平方和变化的程度

如下是前列腺癌的例子分析结果:

Term	Coefficient	Std. Error
Intercept	2.46	0.09
lcavol	0.68	0.13
lweight	0.26	0.10
age	−0.14	0.10
lbph	0.21	0.10
svi	0.31	0.12
lcp	−0.29	0.15
gleason	−0.02	0.15
pgg45	0.27	0.15

根据上述内容可以算出每一个参数对应的置信区间，发现参数 age、lcp、gleason、pgg45 的置信区间包含 0，即认为其影响可能不显著。那么这其实就是一个假设检验问题了，我们想要看 4 个变量的影响是否显著，于是删除 4 个变量，原假设 H_0：认为 4 个变量都是不显著的

F 统计量为

$$F=\frac{(32.81-29.43)/(8-4)}{29.43/(67-8-1)}=1.67$$

而在 $F_{4,58}$ 时大于此时 F 统计量的概率为:

$$\Pr(F_{4,58}>1.67)=0.17$$

p 检验越小越拒绝，$0.17>0.05$ 因此接受原假设 H_0，认为这四个变量都是不显著的。

2.2 岭回归（Ridge Regression）

上述问题在 $N < p+1$ 时由于 X^TX 不可逆，会发生过拟合，因此加上 λ 倍的单位矩阵 I：

$$\hat{\theta} = (X^TX + \lambda I)^{-1}X^Ty$$

因此 Ridge 回归优化的结构风险：

$$\mathrm{RSS}^{\mathrm{Ridge}}(\theta) = \sum_{i=1}^{N}\left(y_i - \theta_0 - \sum_{j=1}^{p}\theta_j x_i^{(j)}\right)^2 + \lambda\sum_{j=1}^{p}\theta_j^2$$

其中 $\lambda\sum_{j=1}^{p}\theta_j^2$ 用于衡量模型复杂度
此时即为求解最优化问题：

$$\hat{\theta}^{\mathrm{Ridge}} = \arg\min_{\theta}\left\{\sum_{i=1}^{N}\left(y_i - \theta_0 - \sum_{j=1}^{p}\theta_j x_i^{(j)}\right)^2 + \lambda\sum_{j=1}^{p}\theta_j^2\right\}$$

λ 是控制收缩范围的参数，用于控制惩罚强度，值越大收缩程度越大，使得这些系数向 0 收缩。平方和加惩罚项也用于神经网络，称为权衰减。

这种直接在残差平方和后加入 L2 正则项，通过 λ 控制惩罚强度属于**无约束的正则化**。由此需要加入约束条件来进一步限制回归系数，缩小假设空间，使 θ_j 不变得特别大：

$$\hat{\theta}^{\mathrm{Ridge}} = \arg\min_{\theta}\left\{\sum_{i=1}^{N}\left(y_i - \theta_0 - \sum_{j=1}^{p}\theta_j x_i^{(j)}\right)^2\right\}$$

$$\text{约束条件：}\ \sum_{j=1}^{p}\theta_j^2 \le t \quad \lambda\text{和}t\text{存在一一对应关系}$$

这里的约束条件由于是平方和相加，我们可以在几何层面将其理解为一个“圆”，t 越小，约束 λ 越大。其中，横纵坐标是回归系数，**蓝色圆形**代表约束条件，**红色椭圆**代表残差平方和（RSS）的等高线（因

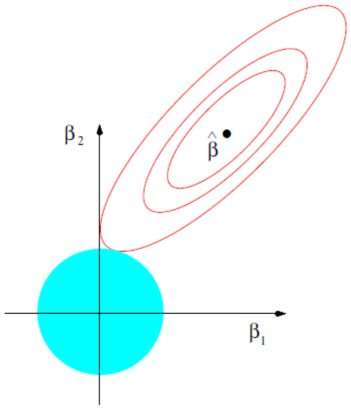


图 1: 岭回归约束优化几何图

为 RSS 是平方型且一个 RSS 有多个解，所有解构成图形为“椭圆型”），**中心** $\hat{\beta}$ 是普通线性回归的解，直接使用就是很容易导致过拟合。由此，我们选取尽可能靠近中心的椭圆，而且能交集蓝色圆的点/切点

注意：

- 求解 Ridge 回归前要标准化输入
- 截距 θ_0 不加入惩罚项，因为加入后会导致 Ridge 回归过程依赖于 y 选取的原点，对 θ_0 的惩罚可能会导致其他回归系数 θ_j 变大
- 用 y 的均值来估计 θ_0，由此输入矩阵 X 是变成了 p 列而不再是一般回归的 $p+1$ 列了

那么接下来的问题就是求解：

$$\hat{\theta}^{\text{Ridge}} = \arg\min_{\theta} \left\{ \sum_{i=1}^{N} \left(y_i - \theta_0 - \sum_{j=1}^{p} \theta_j x_i^{(j)} \right)^2 + \lambda \sum_{j=1}^{p} \theta_j^2 \right\}$$

表示成矩阵形式为：

$$\hat{\theta}^{\text{Ridge}} = \arg\min_{\theta} (\boldsymbol{y} - \boldsymbol{X}\theta)^T (\boldsymbol{y} - \boldsymbol{X}\theta) + \lambda \theta^T \theta$$

注意，这里的 θ 向量元素是从 θ_1 开始的，不包含 θ_0！

岭回归的解为：

$$\hat{\theta}^{\text{Ridge}} = (\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

那么岭回归是如何对模型复杂度进行“收缩”的呢？于是引出了有效自由度（有效参数个数）来衡量正则化后模型的实际复杂度，用来表示正则化后模型“真正用到”的参数数量:

$$\text{df}(\lambda) = \text{trace}\left[(X^T X + \lambda I)^{-1} X^T X \right] = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}$$

（1）其中 d_j^2 是 $X^T X$ 的第 j 个特征值（2）d_j^2 越小，有效参数个数减少的幅度越大

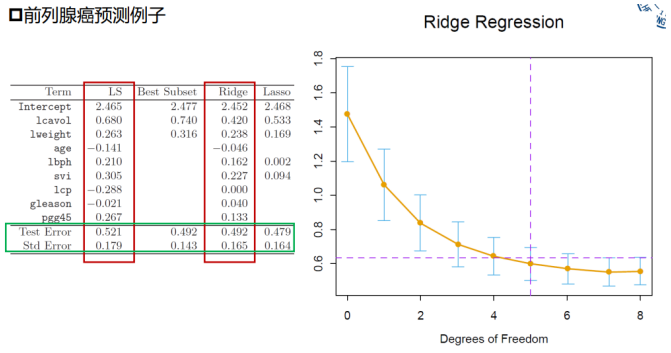
（3）λ 越大，有效参数个数减少的幅度越大（4）$\lambda = 0$ 则退化成最小二乘，相当于没有惩罚约束

（5）$\lambda = 0$ 趋近于无穷时，$df(\lambda)$ 趋近 0，相当于所有参数都被收缩至 0

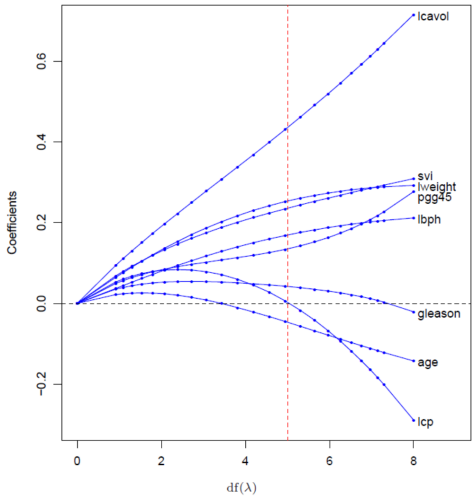
如下是前列腺癌数据集的分析：

□前列腺癌预测例子

Term	LS	Best Subset	Ridge	Lasso
Intercept	2.465	2.477	2.452	2.468
lcavol	0.680	0.740	0.420	0.533
lweight	0.263	0.316	0.238	0.169
age	−0.141		−0.046	
lbph	0.210		0.162	0.002
svi	0.305		0.227	0.094
lcp	−0.288		0.000	
gleason	−0.021		0.040	
pgg45	0.267		0.133	
Test Error	0.521	0.492	0.492	0.479
Std Error	0.179	0.143	0.165	0.164



上图纵坐标是交叉验证误差，可见随着有效参数个数增大，交叉验证误差减小，在 8 个参数时有最小。但是其实可以看见有效参数个数下降到 5 个时，误差上升的并不多，可以达到减少参数个数而几乎不影响误差的效果。



而这个图所展示的是随着有效自由度的变化，不同变量的特征系数变化情况，可见在有效参数个数为 5 时能够降低特征系数，且部分变量的特征系数趋近 0.

2.3 LASSO 回归

LASSO 回归与岭回归相似，同样地增添了正则项来衡量模型复杂度，进行惩罚。

结构风险为：

$$\mathrm{RSS}^{\mathrm{LASSO}}(\theta)=\sum_{i=1}^{N}\left(y_i-\theta_0-\sum_{j=1}^{p}\theta_j x_i^{(j)}\right)^2+\lambda\sum_{j=1}^{p}|\theta_j|$$

求解最优化问题为：

$$\hat{\theta}^{\mathrm{LASSO}}=\arg\min_{\theta}\left\{\sum_{i=1}^{N}\left(y_i-\theta_0-\sum_{j=1}^{p}\theta_j x_i^{(j)}\right)^2+\lambda\sum_{j=1}^{p}|\theta_j|\right\}$$

对比岭回归的正则项核心是 θ_j^2，LASSO 选取的是 $|\theta_j|$；λ 越大，收缩的程度越大。这些系数向 0 收缩，向 0 收缩的程度大于岭回归。

同样的思路，LASSO 也可以**对惩罚项进行约束**，等价于一个有条件最优化问题：

$$\hat{\theta}^{\mathrm{LASSO}}=\arg\min_{\theta}\left\{\sum_{i=1}^{N}\left(y_i-\theta_0-\sum_{j=1}^{p}\theta_j x_i^{(j)}\right)^2\right\}$$

$$\text{约束条件：}\sum_{j=1}^{p}|\theta_j|\le s\quad \lambda\text{和}s\text{存在一一对应关系}$$

s 越小，约束越大。LASSO 的约束使得解对于 y 非线性，即 y 变化时，有些系数可能变为 0；LASSO 没有闭式解，本质是二次规划问题，这是由于 LASSO 选取的是绝对值进行惩罚，正则项在 0 处不可导，不能像岭回归一样对 θ 偏导等于 0 解出 $\hat{\theta}$。

同样地，可以进行几何的直观解释：

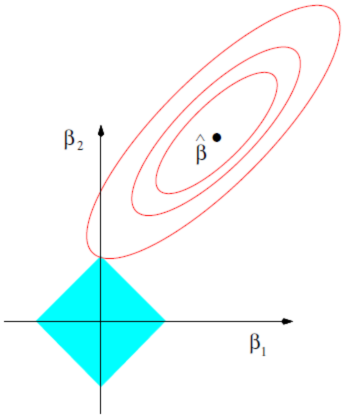
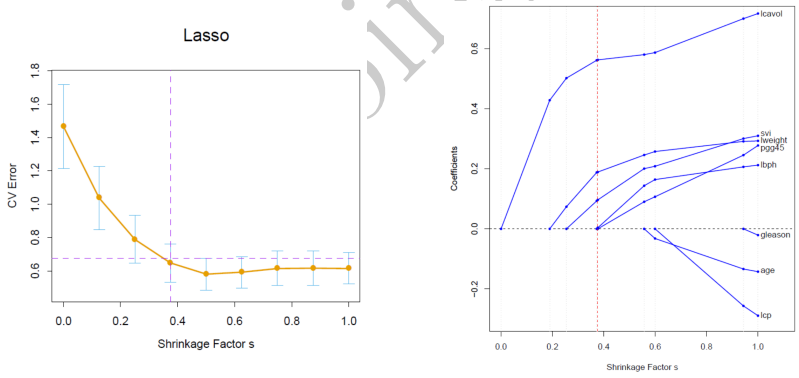


图 2: LASSO 回归约束优化几何图

约束范围成“菱形”是因为选取的绝对值型约束，LASSO 的约束易切于顶点，这样让部分系数直接归 0.

LASSO 对前列腺癌的例子分析结果如下：



可见随着 s 值变小，约束增大时，在一定程度上 CV Error 几乎无变化，但是几个变量的参数已经收缩到 0，可以将其剔除。

2.4 弹性网及拓展

拓展：

对比岭回归和 LASSO 的正则项，我们可以对其进行拓展，即把原本的 L1 和 L2 正则项延伸到 Lq 正则项，用 $\lambda\sum_{j=1}^{p}|\theta_j|^q$ 来代表惩罚项，由此不同的 q 值对应不同的约束形状。

$$\mathrm{RSS}^q(\theta)=\sum_{i=1}^{N}\left(y_i-\theta_0-\sum_{j=1}^{p}\theta_j x_i^{(j)}\right)^2+\lambda\sum_{j=1}^{p}|\theta_j|^q$$

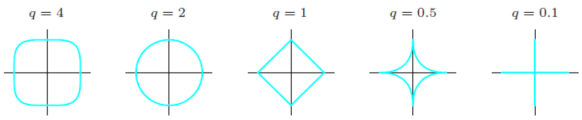


FIGURE 3.12. *Contours of constant value of* $\sum_j |\beta_j|^q$ *for given values of* q.

图 3: 不同 q 值下的约束形状

弹性网:

同时结合了 Ridge 和 LASSO 的惩罚项, 其约束图形是 “圆角菱形”，其既可以让系数为 0，又能平滑处理共线性。

$$\mathrm{RSS}^{\mathrm{Elastic}}(\theta) = \sum_{i=1}^{N} \left(y_i - \theta_0 - \sum_{j=1}^{p} \theta_j x_i^{(j)} \right)^2 + \lambda \sum_{j=1}^{p} \left(\alpha \theta_j^2 + (1-\alpha)|\theta_j| \right)$$

2.5 偏差-方差分解

- 偏差（Bias）：估计出来的模型和真实模型之间的平均距离
- 方差（Variance）：不同训练子集训练出的模型，在同一测试样本上预测结果的波动程度，反映模型之间的平均差距。模型越复杂，对训练数据的微小变化越敏感。

如下是训练误差和测试误差所反映的方差与偏差图:

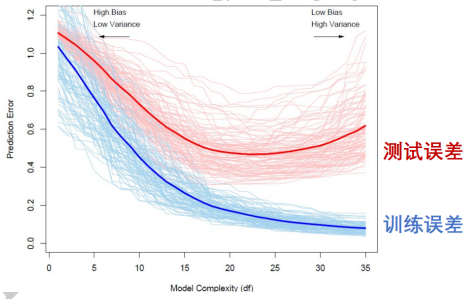


图 4: 偏差-方差图

假设真实的数据模型是:

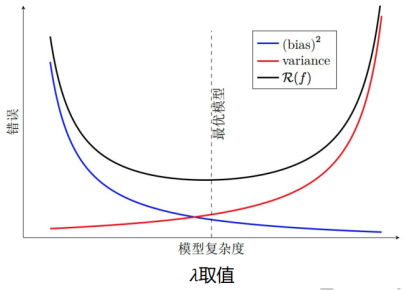
$$Y = f(X) + \varepsilon$$

其中 ε 是噪声，有 $\mathbb{E}(\varepsilon) = 0$, $\mathrm{Var}(\varepsilon) = \sigma^2$；考虑平方损失函数下的最小二乘线性回归函数拟合 $Y = \hat{f}(X)$；所以有在输入点 $X = x_0$ 的期望预测误差:

$$
\begin{aligned}
\mathrm{Err}(x_0) &= \mathbb{E}_{\text{训练集}}\left[\left(Y-\hat{f}(X)\right)^2 \mid X=x_0\right] \\
&= \sigma^2+\left[\mathbb{E}_{\text{训练集}}\left[\hat{f}(x_0)\right]-f(x_0)\right]^2+\mathbb{E}_{\text{训练集}}\left[\hat{f}(x_0)-\mathbb{E}_{\text{训练集}}\left[\hat{f}(x_0)\right]\right]^2 \\
&= \sigma^2+\mathrm{Bias}^2\left(\hat{f}(x_0)\right)+\mathrm{Var}\left(\hat{f}(x_0)\right) \\
&= \text{不可约误差}+\text{偏差}^2+\text{方差}
\end{aligned}
$$

模型 $\hat{f}$ 越复杂，参数越多，偏差越小，方差越大

下图则可展示模型复杂度、偏差、方差和总体期望测试误差（$R(f)$）的关系；λ 越大，模型复杂度越低。



Ridge 与 LASSO 模型的评选选取**交叉验证**，希望找到最小化期望预测误差的、合适的 λ.

如果有**充足的数据**，最好的解决方案是将数据集随机分成三个部分:

- 训练集：用于拟合模型
- 验证集：估计预测误差进行模型选择
- 测试集: 评估最终模型的泛化能力

注意：测试集应该放在一边，模型选择和训练都不能用到，仅仅在最后模型测试阶段才使用；一般来说，50% 的数据于用训练，25% 用于验证，剩下 25% 用于测试。

但是，往往数据量是有限且稀少的，如果直接划分训练集和测试集，会浪费宝贵的数据资源，且评估结果可能不稳定。因此，选择 **k 折交叉验证**来解决这个问题，通过手头数据的一部分来训练模型，用另外一部分测试:

- 把数据分割成粗略等大小的 K 部分，常选取 K=5 或 10，K=N（样本总数）则称为留一交叉验证
- 我们依次选取第 k 部分作为测试集，其他 K-1 部分作为训练集，计算每次训练模型在第 k 部分的预测结果和预测误差
- 对 k=1,2,3⋯,K 进行操作，然后将 k 个预测误差的估计组合起来

$\hat{f}^{-i}(x)$ 表示移去第 i 部分样本后，在剩余的 K-1 部分样本上计算获得的模型，该模型计算第 i 部分

样本中的平均测试误差：

$$e_i = \frac{1}{\text{第}i\text{部分样本数目}} \sum_{j \text{ 属于第}i\text{部分}} L\left(y_j, \hat{f}^{-i}(x_j)\right)$$

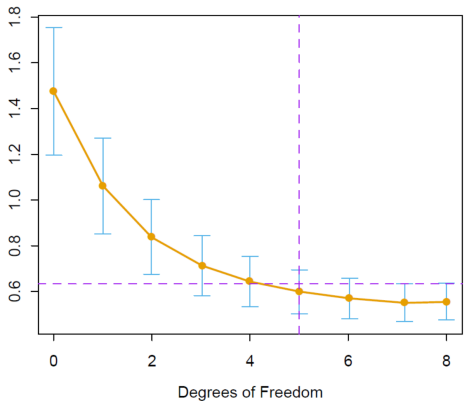
计算 e_i 的均值为 CV 误差均值：

$$\text{AveErr}_{\text{CV}} = \frac{1}{K}\sum_{i=1}^{K} e_i$$

e_i 的标准误为 CV 的 $s.e.m = std/sqrt(K)$：

$$\text{SEM}_{\text{CV}} = \sqrt{\frac{1}{K}\sqrt{\frac{1}{K}\sum_{i=1}^{K}(e_i - \text{AveErr}_{\text{CV}})^2}}$$

考虑在 LASSO 和岭回归的交叉验证，我们可以对每个 λ 计算 CV 误差均值和 s.e.m，由此选择最优参数 λ：



3 线性分类

3.1 线性判别函数

先前我们的线性回归是拟合后的模型来预测测试集，得到的是预测的数值 y，而此处的线性判别函数则是对结果进行一个**分类**，那么预测的是**类别**。

数据：

- 输入向量 $x=\left(x^{(1)},x^{(2)},\ldots,x^{(p)}\right)^T$（取为定量的输入值 $x_1,x_2\ldots$ 之类的，或者是其非线性变换）
- 输出标量 y，代表结果的类别。如最简单的二分法，就是把事物分成 2 类，$y=1$ 代表一类（Ω_1），$y=-1$ 代表另一类（Ω_2）.
- 数据集 $\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$

模型：

线性判别函数与线性回归类似：

$$g(x)=\theta_0+\theta_1x^{(1)}+\theta_2x^{(2)}+\cdots+\theta_px^{(p)}=\theta_0+\theta^Tx$$

$$\text{决策规则}=\begin{cases}\text{如果}g(x)>0,\ \text{则判别}x\in\Omega_1\\ \text{如果}g(x)<0,\ \text{则判别}x\in\Omega_2\\ \text{如果}g(x)=0,\ \text{则将}x\ \text{任意分到一类}\end{cases}$$

θ_0 和$\theta=(\theta_1,\ldots,\theta_p)^T$ 是待学习的参数向量。θ 为权向量，θ_0 为阈值权（相当于基准线，调整分类的边界）。

决策面：判别函数可以写成：$g(x)=\theta_0+\theta^Tx$，而回顾数学中平面方程为：$Ax+By+Cz+D=0$，其中法向量为：$\vec{n}=(A,B,C)$。那么同理，当判别函数等于 0 时，相当于一个“平面”把两个分类 Ω_1 和 Ω_2 分开，把这个“平面”称为超平面 H，分割开的**决策区域**是 $\mathcal{R}_1:g>0$ 和 $\mathcal{R}_2:g<0$；有以下结果：

超平面法向量为：$\vec{w}=\theta^T$，　　法向量的单位向量为：$\hat{w}=\frac{\vec{w}}{\|\vec{w}\|}$，　　其中 $\|\cdot\|$ 指向量的范数（长度）

由此，具体点有 $g(\vec{x})$，假设其到超平面的投影为 P，点 P 到点 x 的向量有 $\vec{d}$，所以：

$$\begin{aligned}g(\vec{x})&=\theta_0+\theta^T\vec{x}=\theta_0+\theta^T(\vec{x}_P+\vec{d})\overset{P\in H}{=}\theta^T\vec{d}\\&=\theta^Td\frac{\vec{w}}{\|\vec{w}\|}=\vec{w}d\frac{\vec{w}}{\|\vec{w}\|}=d\|\vec{w}\|\\&\Rightarrow d=\frac{g(\vec{x})}{\|\vec{w}\|}\end{aligned}$$

得到的具体点 x 到超平面的代数距离为：$d=\frac{g(\vec{x})}{\|\vec{w}\|}$，几何距离为：$d=\frac{|g(\vec{x})|}{\|\vec{w}\|}$

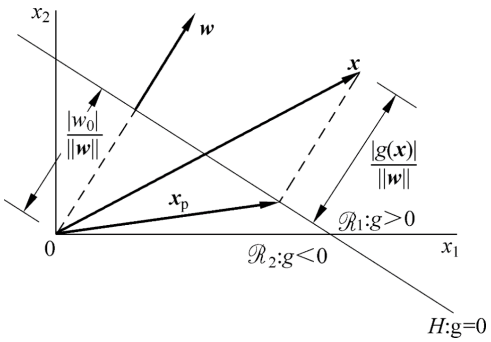


图 5: 决策面示意图

3.2 Fisher 线性判别

线性判别分析（LDA），其目的是将样本投影到一维空间上，让不同类别样本尽可能分开，同类样本尽可能聚集；因此希望找到一个最优的投影方向向量 θ，使得**投影后的样本值**为：

$$y_i = \theta^T x_i + \theta_0, \quad i = 1, 2, \ldots, N$$

注意：这里的 y_i 仅是投影值，要区别于线性回归的预测输出值 y_i，两者没有关联

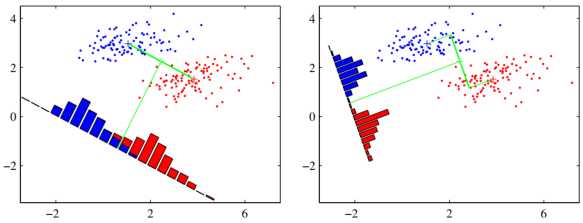


图 6: 将样本进行不同方向的投影

投影前的 p 维空间：

类内离散度矩阵（衡量一个类中的离散程度）：

$$\boldsymbol{S}_i = \sum_{x_j \in \Omega_i} (x_j - m_i)(x_j - m_i)^T, \quad i = 1, 2$$

此处注意辨析一下 x 相关的向量：我们总的 x 数据是一个 $n \times p$ 的矩阵：$X = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1p} \\ x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{np} \end{pmatrix}$

x_j 代表第 j 行数据：$x_j = \left(x_j^{(1)}, x_j^{(2)}, \ldots, x_j^{(p)}\right)^T$

其中，**类均值向量**为：

$$m_i = \frac{1}{N_i} \sum_{x_j \in \Omega_i} x_j, \quad i = 1, 2$$

根据上式可知类内离散度矩阵是一个 $p\times p$ 矩阵，两个类别合并起来有**总类内离散度矩阵**：$\boldsymbol{S}_w = \boldsymbol{S}_1 + \boldsymbol{S}_2$；而两个类别之间有**类间离散度矩阵**：

$$\boldsymbol{S}_b = (m_1 - m_2)(m_1 - m_2)^T, \quad i = 1, 2$$

投影后的一维空间

上述我们将这些离散点的**类内**和**类间**的离散度表示了出来，但表达的方式均是矩阵，我们要让类内离散度越小、类间离散度越大，则需要**降维**说明，因此选取投影到一维空间中，这个一维的向量即为超平面的法向量。投影后有：

类内离散度：

$$\tilde{S}_i^2 = \sum_{y_j \in \Omega_i} (y_j - \tilde{m}_i)^2, \quad i = 1, 2$$

其中，类均值向量为：$\tilde{m}_i = \frac{1}{N_i}\sum_{y_j\in\Omega_i} y_j = \frac{1}{N_i}\sum_{x_j\in\Omega_i} \theta^T x_j + \theta_0 = \theta^T m_i + \theta_0, \quad i = 1, 2$

总类内离散度：$\tilde{S}_w = \tilde{S}_1^2 + \tilde{S}_2^2$

类间离散度：

$$\tilde{S}_b = (\tilde{m}_1 - \tilde{m}_2)^2, \quad i = 1, 2$$

那么我们想要使投影后两类尽可能分开，而各类内部尽可能聚集：

$$\max_\theta J_F(\theta) = \frac{\tilde{S}_b}{\tilde{S}_w} = \frac{(\tilde{m}_1 - \tilde{m}_2)^2}{\tilde{S}_1^2 + \tilde{S}_2^2}$$

上式推导得到广义 Rayleigh 商：

$$\max_\theta J_F(\theta) = \frac{\theta^T \boldsymbol{S}_b \theta}{\theta^T \boldsymbol{S}_w \theta}$$

因此转换为等价问题：

$$\max_\theta \ \theta^T \boldsymbol{S}_b \theta$$
$$\text{s.t. } \theta^T \boldsymbol{S}_w \theta = c$$

为什么要/可以把 $\theta^T \boldsymbol{S}_w \theta$ 等于某个数 c 呢?

这是因为我们的目的是想算得最优的 θ 投影方向，而并非关注其方向向量的大小，θ 和 $k\theta$ 都是代表一个方向。所以我们就限制 $\theta^T \boldsymbol{S}_w \theta$ 的值，在 θ 的大小约束下，找到最优的方向即可。

推导过程如下：

$$\widetilde{S}_b = (\widetilde{m}_1 - \widetilde{m}_2)^2 = (\theta^T m_1 + \theta_0 - \theta^T m_2 - \theta_0)^2 = (\theta^T m_1 - \theta^T m_2)^2 = \theta^T (m_1 - m_2)(m_1 - m_2)^T \theta = \theta^T S_b \theta$$

$$
\begin{aligned}
\widetilde{S}_w &= \widetilde{S}_1^2 + \widetilde{S}_2^2 = \sum_{y_i \in \Omega_1} (y_i - \widetilde{m}_1)^2 + \sum_{y_j \in \Omega_2} (y_j - \widetilde{m}_2)^2 \\
&= \sum_{x_i \in \Omega_1} (\theta^T x_i + \theta_0 - \theta^T m_1 - \theta_0)^2 + \sum_{x_j \in \Omega_2} (\theta^T x_j + \theta_0 - \theta^T m_2 - \theta_0)^2 \\
&= \sum_{x_i \in \Omega_1} (\theta^T x_i - \theta^T m_1)^2 + \sum_{x_j \in \Omega_2} (\theta^T x_j - \theta^T m_2)^2 \\
&= \sum_{x_i \in \Omega_1} \theta^T (x_i - m_1)(x_i - m_1)^T \theta + \sum_{x_j \in \Omega_2} \theta^T (x_j - m_2)(x_j - m_2)^T \theta \\
&= \theta^T S_1 \theta + \theta^T S_2 \theta = \theta^T S_w \theta
\end{aligned}
$$

那么对于一个约束问题我们选取微积分 II 中的**拉格朗日乘数法**去求解：

拉格朗日函数：

$$L(\theta, \lambda) = \theta^T \boldsymbol{S}_b \theta - \lambda \left(\theta^T \boldsymbol{S}_w \theta - c \right)$$

求 θ 偏导，$\frac{\partial L(\theta,\lambda)}{\partial \theta} = 0$，极值解 θ^* 满足：

$$\boldsymbol{S}_b \theta^* - \lambda \boldsymbol{S}_w \theta^* = 0$$

假设 $\boldsymbol{S}_w$ 是满秩的，可得：

$$\boldsymbol{S}_w^{-1} \boldsymbol{S}_b \theta^* = \lambda \theta^*$$

带入 $\boldsymbol{S}_b$ 有：

$$\lambda \theta^* = \boldsymbol{S}_w^{-1} (m_1 - m_2)(m_1 - m_2)^T \theta^*$$

$(m_1 - m_2)^T$ 是 $1 \times p$ 的矩阵，θ^* 是 $p \times 1$ 的矩阵，所以 $(m_1 - m_2)^T \theta^*$ 为标量，可以记成 k' ，所以不考虑 θ^* 大小，仅考虑方向时，得到：

$$\theta^* = \boldsymbol{S}_w^{-1} (m_1 - m_2)$$

所以，最后的决策规则：

$$\text{若} g(x) = \theta^{*T} x + \theta_0 \begin{cases} > 0, & \text{则} x \in \Omega_1 \\ < 0, & \text{则} x \in \Omega_2 \end{cases}$$

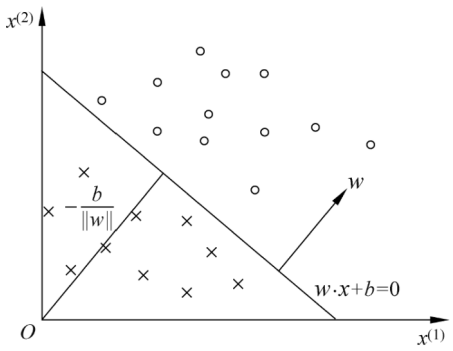
确定阈值 θ_0，依据经验有两种选择方式：

$$\theta_0 = -\frac{1}{2} (\widetilde{m}_1 + \widetilde{m}_2) \quad \theta_0 = -\widetilde{m} (\widetilde{m} \text{ 是所有样本在投影后的均值})$$

3.3 感知机

先前的 Fisher 线性判别是通过降维、离散度寻找最优的投影方向；而这里的感知机是一个二分类的线性分类模型。它的输入是一个样本的特征向量，输出是这个样本所属的类别（通常用 +1 和 -1 表示）。它的目标是在特征空间中找到一个线性超平面，将不同类别的样本完全分开。

感知机的数据与先前一致，其模型本质还是定义一个分离超平面。但数据一定要线性可分的，这样才能有限次迭代找到一个能完全分开两类的超平面。



$$g(x) = \text{sign}\left(\theta_0 + \theta^T x\right)$$

$$\text{sign}(z) = \begin{cases} +1, & z \geqslant 0 \\ -1, & z < 0 \end{cases}$$

同样的，输入向量是 $x = \left(x^{(1)}, x^{(2)}, \ldots, x^{(p)}\right)^T$。

感知机开始时的超平面无法把数据完全正确地分成两个类别，所以会有样本被分到错误一侧，这些就是**误分类点**。比如点 i 的标签是 $y_i = 1$，但模型计算出来的 $\theta_0 + \theta^T x$ 符号为负，就把这个点错误地分到标签为 -1 的一类去了。

误分类点满足：

$$-y_i\left(\theta_0 + \theta^T x_i\right) > 0$$

误分类点到超平面的距离：

$$-\frac{1}{\|\theta\|} y_i\left(\theta_0 + \theta^T x_i\right)$$

所有误分类点总距离：

$$-\frac{1}{\|\theta\|} \sum_{x_i \in \text{误分类点集合} M} y_i\left(\theta_0 + \theta^T x_i\right)$$

只要训练集是线性可分的，那么只要一直调整超平面，最后会是没有误分类点的存在，此时的所有误分类点总距离为 0. 因此把问题转换成最小化损失函数：

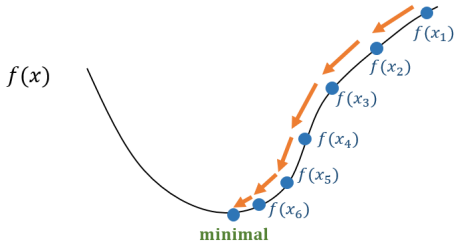
$$\min_{\theta, \theta_0} L(\theta, \theta_0) = -\sum_{x_i \in M} y_i\left(\theta_0 + \theta^T x_i\right)$$

要求解这个无约束最优问题，选取**梯度下降法**，如：求解 $\min_x f(x)$，如果我们能够构造一个序列 $x^0, x^1, x^2, \ldots$ 并满足 $f(x^{t+1}) < f(x^t), \quad t = 0, 1, 2, \ldots$，那么不断进行下去，我们就可以不断逼近局部的极小值。

我们可以看成每一次处理都是一小步 Δx，根据泰勒展开：

$$f(x+\Delta x) \approx f(x) + \Delta x \nabla f(x)$$

$\nabla f(x) = \frac{df(x)}{dx}$ 为梯度方向（函数增长方向），如下图即表示 $\Delta x < 0$ 时不断下降的过程。



可以令 $\Delta x = -\alpha \nabla f(x)$，其中步长 $\alpha > 0$ 是一个较小的正数，从而：$\Delta x \nabla f(x) = -\alpha \left(\nabla f(x)\right)^2 < 0$（为什么要选取 $\Delta x = -\alpha \nabla f(x)$ 呢，这是让下降方向始终为梯度方向，沿负梯度方向下降最快，更加快速高效）

所以回到我们的感知机：

求解 $\min_{\theta,\theta_0} L(\theta,\theta_0) = -\sum_{x_i \in M} y_i \left(\theta_0 + \theta^T x_i\right)$ 选取随机梯度下降法

首先任意选取一个超平面，有 θ 和 θ_0，损失函数的梯度为：

$$\nabla_\theta L(\theta,\theta_0) = -\sum_{x_i \in M} y_i x_i$$

$$\nabla_{\theta_0} L(\theta,\theta_0) = -\sum_{x_i \in M} y_i$$

随机选取一个误分类点更新参数（$0 < \eta \leq 1$ 是步长，也称为学习率）：

$$\theta \leftarrow \theta + \eta y_i x_i$$

$$\theta_0 \leftarrow \theta_0 + \eta y_i$$

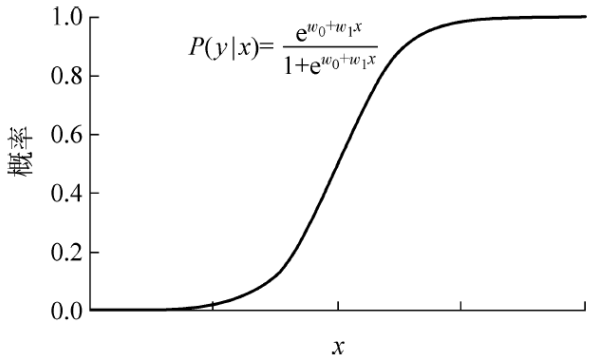
3.4 罗杰斯特回归（Logistic Regression）

罗杰斯特回归（Logistic Regression）本质是线性模型，通过 Sigmoid 函数将输出映射为 [0, 1] 概率值，根据概率值来判断样本属于某一类别的可能性。

考虑简单的一维输入情况：$x = x^{(1)}$，输出值 y 是我们分类的标签 1 或-1，类别概率建模：$P(y = 1|x) = f(x)$

经验发现，”S” 型函数适合建模这种概率：

$$P(y|x) = f(x) = \frac{e^{\theta_0 + \theta_1 x}}{1 + e^{\theta_0 + \theta_1 x}}$$



在神经网络中，这个罗杰斯特函数又被称为 Sigmoid 函数：

$$\pi(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$$

$\pi(s)$ 满足 $\pi(-s) = 1 - \pi(s)$

那么可以将模型预测出来的标签为 1 和-1 的概率进行作比，叫做**几率（odds）**：

$$\frac{P(y=1|x)}{P(y=-1|x)} = \frac{\frac{1}{1+e^{-s}}}{\frac{e^{-s}}{1+e^{-s}}} = e^s = e^{\theta_0+\theta_1 x}$$

取对数为**对数几率（log odds）**：

$$\ln\left(\frac{P(y|x)}{1-P(y|x)}\right) = \theta_0 + \theta_1 x$$

logit 函数：

$$\text{logit}\big(P(y|x)\big) = \ln\left(\frac{P(y|x)}{1-P(y|x)}\right)$$

那么对于我们的多维数据：

$$\text{logit}\big(P(y|x)\big) = \ln\left(\frac{P(y|x)}{1-P(y|x)}\right) = \theta_0 + \theta_1 x^{(1)} + \cdots + \theta_p x^{(p)} = \theta_0 + \theta^T x$$

$$P(y|x) = \frac{e^{\theta_0+\theta^T x}}{1+e^{\theta_0+\theta^T x}}$$

y=1 的概率大于 y=-1 的概率时，则认为分类到 $y=1$ 类，有如下决策规则：

$$\begin{cases} \text{若logit}\big(P(y|x)\big) > 0, & \text{则} x \in \Omega_1 \\ \text{若logit}\big(P(y|x)\big) < 0, & \text{则} x \in \Omega_2 \end{cases}$$

有了上述的背景铺垫，接下来就是求解 θ 的问题，模型在样本 (x_j, y_j) 上的概率：

$$P(y_j|x_j) = \begin{cases} \pi(\theta^T x_j), & y_j = +1 \\ 1-\pi(\theta^T x_j) = \pi(-\theta^T x_j), & y_j = -1 \end{cases}$$

注意：这里的 θ 是增广的向量 $\theta=(\theta_0,\theta_1,\theta_2,\ldots,\theta_p)$，同样，$x$ 也是增广的向量 $x=(1,x_1,x_2,\ldots,x_p)$

上述这个式子等价为：$P(y_j|x_j)=\pi(y_j\theta^Tx_j)$

对于多个样本的概率，我们要解出最可能的 θ 则可选取概统中的**极大似然估计法（MLE）**计算 θ 的极大似然估计量。

假设样本是独立同分布的，对于所有样本，模型的似然函数是：

$$L(\theta)=\prod_{j=1}^{N}P(y_j|x_j)=\prod_{j=1}^{N}\pi(y_j\theta^Tx_j)$$

似然函数是乘积形式，不宜直接优化，所以取对数转化为求和形式：

$$\max_{\theta}\ln(L(\theta))=\sum_{j=1}^{N}\ln\left(P(y_j|x_j)\right)=\sum_{j=1}^{N}\ln\left(\pi(y_j\theta^Tx_j)\right)$$

考虑到之前的**梯度下降法**我们是求取的最小值，所以可以对这个问题变成最小化问题：

$$\min_{\theta}-\ln(L(\theta))=\sum_{j=1}^{N}\ln\left(\frac{1}{\pi(y_j\theta^Tx_j)}\right)=\sum_{j=1}^{N}\ln\left(1+e^{-y_j\theta^Tx_j}\right)$$

所以最后的最优化问题是：

$$\min_{\theta}J(\theta)=-\ln(L(\theta))=\sum_{j=1}^{N}\ln\left(1+e^{-y_j\theta^Tx_j}\right)$$

步骤：

1）初始化参数 $\theta=0$

2）目标函数的负梯度：

$$\nabla J=-\sum_{j=1}^{N}\frac{y_jx_j}{1+e^{y_j\theta(k)^Tx_j}}$$

学习率更新参数：

$$\theta(k+1)=\theta(k)-\eta\nabla J$$

检查是否达到终止条件，直到终止，输出 θ 值。

4 多层感知器（MLP）

4.1 多层感知器模型

多层感知器（MLP）是一种神经网络模型，其将输入的多个数据维度映射到单一的输出的数据维度上，由输入层、隐藏层、输出层组成。那么为什么要有多层感知器呢？先前我们所学的感知机是一个线性分类的模型，但是对于非线性的问题则无法解决。比如：要判断一张图片中是不是苹果，仅仅判断单个因素——颜色或形状，都是不完善的，我们要综合颜色、形状等多个要素来判断。这就是多层感知器的作用，用于解决**非线性**的模型。从神经元的角度来看，多层感知器就是一个多输入的模型。

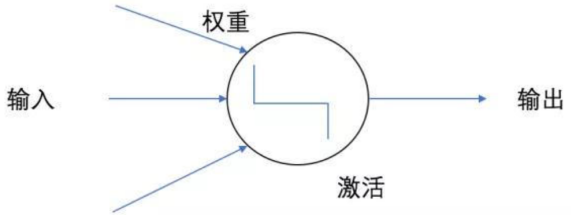


图 7: 神经元示例

模型：对于基本神经元模型，有阈值逻辑单元（TLU）

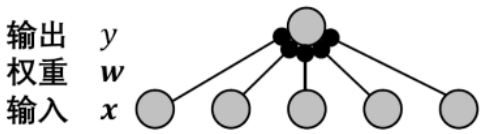


图 8: 基本神经元模型

阈值逻辑单元（TLU）：

$$y = F\left(\sum_{i=1}^{n} w_i x_i + w_0\right) = F\left(w^T x + w_0\right)$$

这里可以想成是先前线性分类的式子：$\theta_0 + \theta^T x$

$F()$ 为“S”形函数，一般为 Sigmoid 函数，称为“激活函数”。

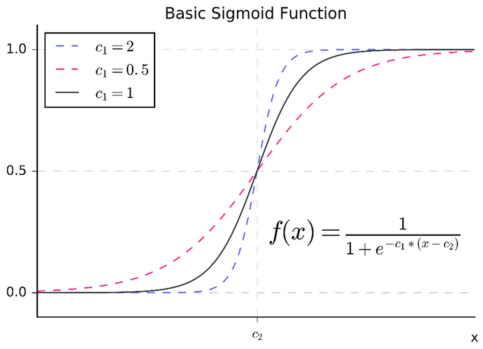


图 9: “S”形函数

感知机：$y = sign(w^T x + w_0)$，$sign()$ 是一种特殊形式的 “S” 函数。由此可以想到，感知机其实是基本神经元模型的一种特殊情况。

双层感知器：上述的模型还停留在单个感知器，在了解多层感知器前，我们先来看看双层感知器的模型。只要隐层神经元数目足够多，就能近似任意连续可微函数。

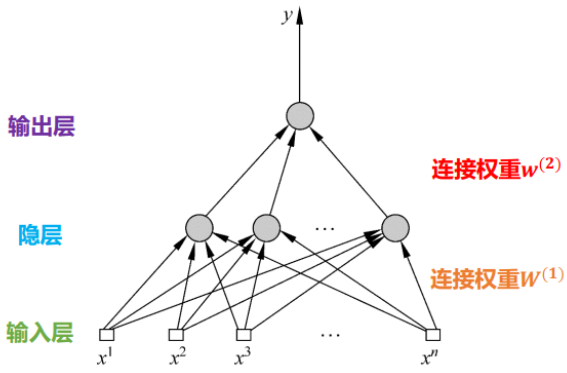


图 10: 双层感知器

那么有模型：

$$y = F\left\{\sum_{j=1}^{n^{(1)}} w_j^{(2)} F\left(\sum_{i=1}^{n^{(0)}} w_{ij}^{(1)} x_i + w_{0j}^{(1)}\right) + w_0^{(2)}\right\}$$

x_i 是输入样本的第 i 个特征值

$n^{(0)}$：输入层神经元数（特征维度）

$n^{(1)}$：隐层神经元数

分解：

$$z_j^{(1)} = \sum_{i=1}^{n^{(0)}} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \quad (j = 1, 2, \ldots, m)$$

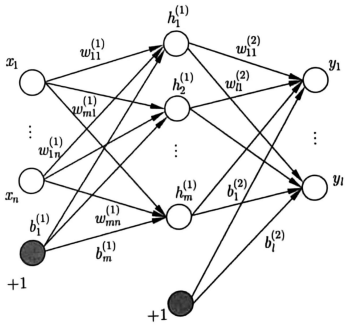
$$h_j^{(1)} = F\left(z_j^{(1)}\right)$$

$$z^{(2)} = \sum_{j=1}^{n^{(1)}} w_j^{(2)} h_j^{(1)} + w_0^{(2)}$$

$$h^{(2)} = F\left(z^{(2)}\right) \; y = h^{(2)}$$

$z^{(1)}$ 代表第一层的净输入；$h^{(1)}$ 代表第一层的输出。

转换为矩阵形式：$h^{(1)} = F\left(z^{(1)}\right) = F\left(\boldsymbol{W}^{(1)T} x + b^{(1)}\right)$；$y = h^{(2)} = F\left(z^{(2)}\right) = F\left(\boldsymbol{W}^{(2)T} h^{(1)} + b^{(2)}\right)$



$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad z^{(1)} = \begin{bmatrix} z_1^{(1)} \\ z_2^{(1)} \\ \vdots \\ z_m^{(1)} \end{bmatrix}, \quad h^{(1)} = \begin{bmatrix} h_1^{(1)} \\ h_2^{(1)} \\ \vdots \\ h_m^{(1)} \end{bmatrix}$$

$$\boldsymbol{W}^{(1)} = \begin{bmatrix} w_{11}^{(1)} & \cdots & w_{1m}^{(1)} \\ \vdots & & \vdots \\ w_{n1}^{(1)} & \cdots & w_{nm}^{(1)} \end{bmatrix}, \quad b^{(1)} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_m^{(1)} \end{bmatrix}$$

多层感知器：多个感知器组成多层级的前馈结构：层间有连接，层内无连接，输入层-隐层-隐层-…-输出层；每个神经元做线性点积 + 非线性激活操作；所有连接权重都是可调节的模型参数；能够实现任意复杂的函数映射。

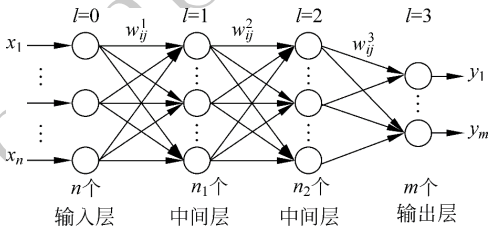
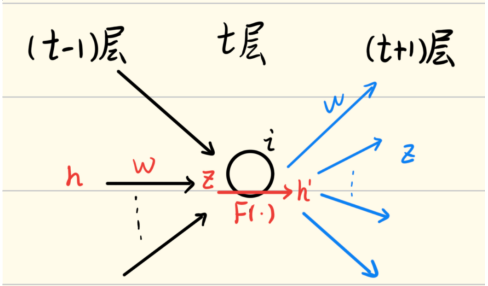


图 11: 多层感知器模型

分解：输入层: $[x_1, x_2, \ldots, x_n]^T$

下图可以帮助着理解隐层的一个神经元的过程变量：



第 t 层的第 j 个神经元：

$$h_j^{(t)} = F\left(z_j^{(t)}\right)$$

$$z_j^{(t)} = \sum_{i=1}^{n^{(t-1)}} w_{ji}^{(t)} h_i^{(t-1)} + w_{j0}^{(t)}$$

输出层的第 k 个神经元：

$$h_k^{(s)} = g\left(z_k^{(s)}\right)$$

$$z_k^{(s)} = \sum_{i=1}^{n^{(s-1)}} w_{ki}^{(s)} h_i^{(s-1)} + w_{k0}^{(s)}$$

矩阵形式：

$$\begin{cases} h^{(1)} = F\left(z^{(1)}\right) = F\left(\boldsymbol{W}^{(1)T} x + b^{(1)}\right) \\ h^{(2)} = F\left(z^{(2)}\right) = F\left(\boldsymbol{W}^{(2)T} h^{(1)} + b^{(2)}\right) \\ \vdots \\ h^{(s-1)} = F\left(z^{(s-1)}\right) = F\left(\boldsymbol{W}^{(s-1)T} h^{(s-2)} + b^{(s-1)}\right) \\ y = h^{(s)} = g\left(z^{(s)}\right) = g\left(\boldsymbol{W}^{(s)T} h^{(s-1)} + b^{(s)}\right) \end{cases}$$

对于多层感知器的**输入层**，往往需要对输入的数据先进行标准化 / 归一化，这是因为如果输入的值范围太大会导致梯度更新偏向数值大的特征，导致模型训练不稳定、收敛慢。

Z-Score 标准化：

$$x_{new} = \frac{x - \mu}{\sigma}$$

其中 μ 是均值，σ 是标准差。

Min-Max 归一化：

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

多层感知器的**隐藏层**是对输入层的特征加权求和后，通过激活函数输出，实现特征的升维/抽象。激活函数的核心作用就是引入**非线性特征**，这样才能达到多层感知器处理非线性数据分布的目的。

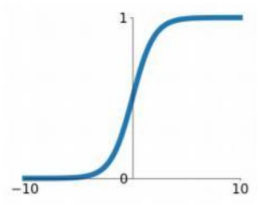
隐层常用激活函数

罗杰斯函数 $\pi()$（Sigmoid）

函数与导数

$$F(z)=\frac{1}{1+e^{-z}}$$

$$F'(z)=F(z)\big(1-F(z)\big)$$

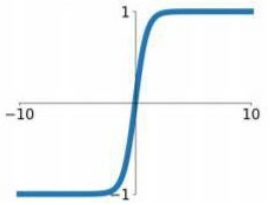


双曲正切函数（tanh）

函数与导数

$$F(z)=\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$$

$$F'(z)=1-F(z)^2$$

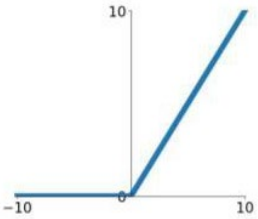


线性整流函数（ReLU）

函数与导数

$$F(z)=\max(0,z)$$

$$F'(z)=\begin{cases}1, & z>0\\ 0, & z\le 0\end{cases}$$



输出层则是输出模型的最终预测结果：

输出层常用函数

回归问题

激活函数是恒等函数

$$y=g\big(z^{(s)}\big)=z^{(s)}$$

$$z^{(s)}=\sum_{i=1}^{n^{(s-1)}}w_i^{(s)}h_i^{(s-1)}+w_0^{(s)}$$

二类分类问题

激活函数是 S 型函数

$$P(y=1|x)=g\big(z^{(s)}\big)=\frac{1}{1+e^{-z^{(s)}}}$$

$$z^{(s)}=\sum_{i=1}^{n^{(s-1)}}w_i^{(s)}h_i^{(s-1)}+w_0^{(s)}$$

多类分类问题

激活函数是软最大化（softmax）函数

$$P(y_k=1|x)=g\big(z_k^{(s)}\big)=\frac{e^{z_k^{(s)}}}{\sum_{i=1}^{K}e^{z_i^{(s)}}}$$

$$z_k^{(s)}=\sum_{j=1}^{n^{(s-1)}}w_{kj}^{(s)}h_j^{(s-1)}+w_{k0}^{(s)}$$

$$k=1,2,\ldots,m$$

4.2 多层感知器的学习算法

多层感知器的模型中，有完整输入输出函数：$f(x;\theta)$；那么我们的 策略 有：

$$\hat{\theta}=\arg\min_{\theta}\left[\sum_{i=1}^{N}L(f(x_i;\theta),y_i)\right]$$

或写为极大似然形式：

$$\hat{\theta}=\arg\min_{\theta}\left[-\sum_{i=1}^{N}\log P(y_i|x_i;\theta)\right]$$

回归问题：损失函数一般取为平方损失

$$\hat{\theta} = \arg\min_{\theta} \left[\sum_{i=1}^{N} (y_i - f(x_i;\theta))^2 \right]$$

二分类问题：只有两种类别，用 0 或 1 表示；损失函数一般取为交叉熵损失；假设 $P(y_i = 1|x_i;\theta) = f(x_i;\theta)$ 遵循伯努利分布

$$\hat{\theta} = \arg\min_{\theta} \left[-\sum_{i=1}^{N} [y_i \log f(x_i;\theta) + (1 - y_i) \log(1 - f(x_i;\theta))] \right]$$

多分类问题：多分类中类别不局限于 2 个，其中我们用 i 表示第 i 个样本，k 表示第 k 个类别；只有当样本属于第 k 个类别时，才有 $y_{ik} = 1$。损失函数一般取为交叉熵损失。假设 $P(y_{ik} = 1|x_i;\theta) = f(x_i;\theta)$ 遵循类别分布

$$\hat{\theta} = \arg\min_{\theta} \left[-\sum_{i=1}^{N} \sum_{k=1}^{l} y_{ik} \log(f(x_i;\theta)) \right]$$

非凸优化问题：无论是 MLP 还是感知机，我们优化的目的都是找到损失函数最小的参数 θ。在感知机的模型中，我们选取了梯度下降法来寻找函数的最小值。

算法：

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{N} \left[\sum_{i=1}^{N} L(f(x_i;\theta), y_i) \right] = \arg\min_{\theta} \frac{1}{N} \left[\sum_{i=1}^{N} L_i(\theta) \right]$$

梯度下降法选取：

$$\theta \leftarrow \theta - \eta \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L_i(\theta)}{\partial \theta}$$

$\eta > 0$ 是学习率；$\frac{\partial L_i(\theta)}{\partial \theta}$ 是第 i 个样本的损失函数梯度向量

但当目标函数存在许多局部极小点时，我们则要使用**随机梯度下降法**。随机梯度下降法：并行计算效率高，避免一些不好的局部极小。一个多层感知器有很多等价参数，所以存在很多等价的局部极小点。

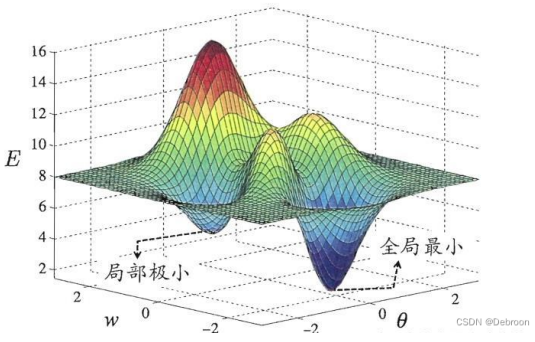


图 12: 非凸优化问题

首先随机打乱样本顺序, 将样本分成 m 个组 (小批量), 每一组有 n 个样本，并随机初始化参数向量。然后针对每组样本，通过以下公式更新参数向量, 并遍历所有样本组，不断迭代，直到收敛为止。

$$\theta \leftarrow \theta - \eta \frac{1}{n} \sum_{j=1}^{n} \frac{\partial L_j(\theta)}{\partial \theta}$$

当 n 是 1 时, 每次参数更新只使用一个样本, 是一种特殊的随机梯度下降。
当 n 是整体样本容量 N 时, 随机梯度下降变为梯度下降

反向传播算法（BP）

基于上述对随机梯度下降法的更新公式，我们的关键就是去计算梯度 $\frac{\partial L_j(\theta)}{\partial \theta}$，但在多层神经网络中，直接求导会非常繁琐低效。因此选取反向传播法，又称误差反向传播，它提供了一种高效的梯度计算以及参数更新方法。

正向传播：基于当前参数重新计算多层感知器中所有变量

反向传播：基于当前变量重新计算对所有参数的梯度

反向传播算法的关键是计算损失函数对各层权重的梯度，由于多层感知机是单个神经元的多层复合，所以梯度计算用**链式法则**求解：

$$\frac{\partial L(w_{ij}^{(1)})}{\partial w_{ij}^{(1)}} = \frac{\partial L(y)}{\partial y} \frac{\partial y}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial h_j^{(1)}} \frac{\partial h_j^{(1)}}{\partial z_j^{(1)}} \frac{\partial z_j^{(1)}}{\partial w_{ij}^{(1)}}$$

反向传播就是链式法则计算的迭代方法：

考虑第 t 个隐层的神经元：

$$h_j^{(t)} = F\left(z_j^{(t)}\right),\ j = 1, 2, \ldots, m$$

$$z_j^{(t)} = \sum_{i=1}^{n^{(t-1)}} w_{ji}^{(t)} h_i^{(t-1)} + w_{j0}^{(t)}$$

第 t + 1 层的神经元：

$$h_k^{(t+1)} = F\left(z_k^{(t+1)}\right),\ k = 1, 2, \ldots, l$$

$$z_k^{(t+1)} = \sum_{j=1}^{n^{(t)}} w_{kj}^{(t+1)} h_j^{(t)} + w_{k0}^{(t+1)}$$

链式法则下，损失函数对第 t 层的参数的梯度是：

$$\frac{\partial L}{\partial w_{ji}^{(t)}} = \frac{\partial L}{\partial z_j^{(t)}} \frac{\partial z_j^{(t)}}{\partial w_{ji}^{(t)}}$$

$$\frac{\partial L}{\partial w_{j0}^{(t)}} = \frac{\partial L}{\partial z_j^{(t)}} \frac{\partial z_j^{(t)}}{\partial w_{j0}^{(t)}}$$

第 t 层的“误差”：即为损失函数对第 t 层净输入的梯度：

$$\delta_j^{(t)} = \frac{\partial L}{\partial z_j^{(t)}}, \quad j = 1, 2, \ldots, n^{(t)}$$

因此代入链式法则有：

$$\frac{\partial L}{\partial w_{ji}^{(t)}} = \delta_j^{(t)} h_i^{(t-1)}$$

$$\frac{\partial L}{\partial w_{j0}^{(t)}} = \delta_j^{(t)}$$

$h_i^{(t-1)}$ 是$t-1$ 层的输出，从正向传播得到

$\delta_j^{(t)}$ 是第t 层的误差，从反向传播得到。

反向传播是指“误差”从输出层到输入层的传递

接下来就是考虑第 t 层的“误差”的反向迭代，依旧使用链式法则：

$$\delta_j^{(t)} = \frac{\partial L}{\partial z_j^{(t)}} = \sum_{k=1}^{n^{(t+1)}} \frac{\partial L}{\partial z_k^{(t+1)}} \frac{\partial z_k^{(t+1)}}{\partial z_j^{(t)}}, \quad j = 1, 2, \ldots, n^{(t)}$$

再将右侧的式子经过函数关系和链式法则拆解：

$$\frac{\partial L}{\partial z_k^{(t+1)}} = \delta_k^{(t+1)}$$

$$\frac{\partial z_k^{(t+1)}}{\partial z_j^{(t)}} = \frac{\partial z_k^{(t+1)}}{\partial h_j^{(t)}} \frac{\partial h_j^{(t)}}{\partial z_j^{(t)}} = w_{kj}^{(t+1)} \frac{dF}{dz_j^{(t)}}$$

得到 $\delta_j^{(t)}$ 的反向迭代关系：

$$\delta_j^{(t)} = \frac{dF}{dz_j^{(t)}} \sum_{k=1}^{n^{(t+1)}} w_{kj}^{(t+1)} \delta_k^{(t+1)}$$

由此我们就得到了每一层之间的误差关系，对于误差迭代的起点——输出层有：$\delta_k^{(s)} = \frac{\partial L}{\partial z_k^{(s)}}$

回归问题有平方损失：$L = \frac{1}{2}\left(h^{(s)} - y\right)^2$，激活函数是恒等函数：$h^{(s)} = g\left(z^{(s)}\right) = z^{(s)}$，由可算得最后一层误差是：

$$\delta^{(s)} = h^{(s)} - y$$

而对于分类问题有交叉熵损失：$L = -\sum_{k=1}^{l} y_k \log h_k^{(s)}$，激活函数是软最大化函数：$h_k^{(s)} = g\left(z_k^{(s)}\right) =$

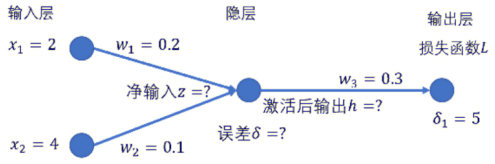
$\frac{e^{z_k^{(s)}}}{\sum_{i=1}^{K} e^{z_i^{(s)}}}$，最后一层误差是：

$$\delta_k^{(s)} = h_k^{(s)} - y_k, \quad k = 1, 2, \ldots, l$$

综上，最后根据正向传播计算的 $h_i^{(t-1)}$ 和反向传播的 $\delta_j^{(t)}$。

回顾上述所有过程，我们的最终目的都是为了算到每个参数的梯度，进而完成梯度下降法对每个参数的迭代，具体应用怎么用可以参考一下作业题 2：

2. 考虑下面的前馈神经网络，激活函数选取为 Sigmoid 函数$F(z) = \frac{1}{1+e^{-z}}$，模型中所有的偏置参数$b = 0$，损失函数为$L(w_1, w_2, w_3)$。利用监督学习对连接权重进行后向传播学习。输入信号、已估计的连接权重和对应的误差如图所示。



（1）请利用前向传播，计算隐层神经元的净输入z以及激活后的输出h。

（2）请利用后向传播，计算隐层的误差δ，之后计算每个连接权重w_1, w_2, w_3的梯度 $\frac{\partial L}{\partial w_1}$，$\frac{\partial L}{\partial w_2}$，$\frac{\partial L}{\partial w_3}$。

（3）最后利用梯度下降法，选取步长$\alpha = 0.2$,计算更新后的连接权重$w_{1,new}, w_{2,new}, w_{3,new}$。

2、(1)、由题：$z = w_1x_1 + w_2x_2 = 0.4 + 0.4 = 0.8$　$h = F(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-0.8}} = 0.690$

(2)、由后向传播的迭代：$\delta_j^{(t)} = \frac{dF}{dz_j^{(t)}} \sum_{k=1}^{q^{(t+1)}} w_{kj}^{(t+1)} \delta_k^{(t+1)}$ $\Rightarrow \delta = -\frac{-e^{-z}}{(1+e^{-z})^2} w_3 \cdot \delta_1 = 0.321$

计算梯度：$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial w_1} = \delta \cdot x_1 = 0.642$

$\frac{\partial L}{\partial w_2} = \delta \cdot x_2 = 1.283$

$\frac{\partial L}{\partial w_3} = \delta_1 \cdot h = 3.45$

(3)、$\alpha = 0.2$.

$\therefore w_{1,new} = w_1 - \alpha \cdot \frac{\partial L}{\partial w_1} = 0.2 - 0.2 \times 0.642 = 0.0716$

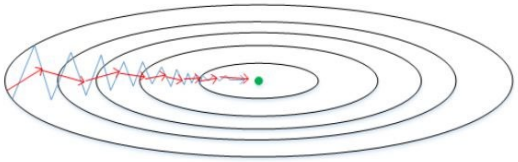
$w_{2,new} = w_2 - \alpha \cdot \frac{\partial L}{\partial w_2} = 0.1 - 0.2 \times 1.283 = -0.1566$

$w_{3,new} = w_3 - \alpha \cdot \frac{\partial L}{\partial w_3} = 0.3 - 0.2 \times 3.45 = -0.39$

4.3 多层感知器的优化算法

先前我们使用的随机梯度下降法（SGD）是让参数沿着梯度的反方向下降，由此不断迭代的过程。但是该方法计算效率低，每一步梯度的更新（方向 + 大小）基于当前所在位置，收敛速度未必最快，所以有多个改进的算法，其中 **Adam** 是最常用的算法。

- **动量（momentum）算法**：动量算法在第 t 步对参数进行如下梯度更新



$$\boldsymbol{v}_t = \beta_1 \boldsymbol{v}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t$$

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \boldsymbol{v}_t$$

其中，θ_t 表示第 t 步的参数，v_t 表示第 t 步的中间变量，g_t 是第 t 步的梯度，β_1 是系数（$0 \leqslant \beta_1 < 1$），η 是学习率。

- **RMSProp 算法**：可认为是对梯度向量归一化

$$\boldsymbol{s}_t = \beta_2 \boldsymbol{s}_{t-1} + (1-\beta_2)\boldsymbol{g}_t \odot \boldsymbol{g}_t$$
$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \frac{1}{\sqrt{\boldsymbol{s}_t + \varepsilon}} \boldsymbol{g}_t$$

其中，s_t 表示第 t 步的中间变量，$\odot$ 表示向量的逐元素积，β_2 是系数（$0 \leqslant \beta_2 < 1$），ε 是每一个元素都为正值的向量，防止分母为 0

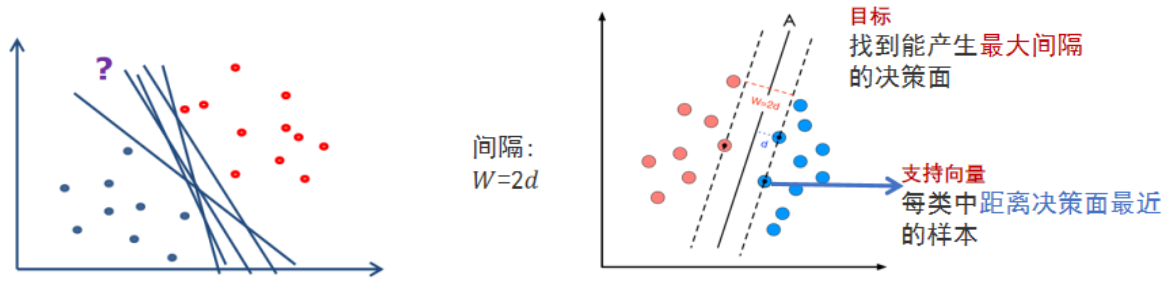
- **Adam 算法**：将动量算法和 RMSProp 算法的技巧结合, 以提高迭代的收敛速度。

$$\boldsymbol{v}_t = \beta_1 \boldsymbol{v}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$$
$$\boldsymbol{s}_t = \beta_2 \boldsymbol{s}_{t-1} + (1-\beta_2)\boldsymbol{g}_t \odot \boldsymbol{g}_t$$
$$\boldsymbol{v}_t = \frac{\boldsymbol{v}_t}{1-\beta_1^t}$$
$$\boldsymbol{s}_t = \frac{\boldsymbol{s}_t}{1-\beta_2^t}$$
$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \frac{\boldsymbol{v}_t}{\sqrt{\boldsymbol{s}_t + \varepsilon}}$$

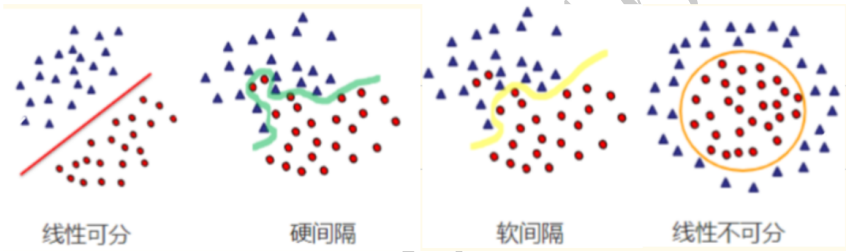
实验证明 Adam 算法对不同的神经网络都有很好的学习收敛速度。

5 支持向量机（SVM）

支持向量机（SVM）是典型的二分类模型，其目标是在特征空间中找到能区分两类样本的边界，并且到最近数据点（支持向量）的**间隔最大**。SVM 在关注“把两类分开”的同时，更关注“分得尽量稳，间隔尽量大”。

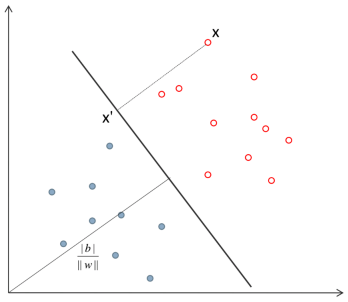


SVM 可分为**线性可分支持向量机**（硬间隔支持向量机）、**线性支持向量机**（软间隔支持向量机）、**非线性支持向量机**。下面展示硬间隔、软间隔、线性不可分的几何图：



5.1 线性可分支持向量机

对于线性可分的数据，我们先前在线性分类的问题中就已经有过讨论（Fisher 线性判别、感知机），如何将两种不同数据分开。而这里则是在分开的同时，还要选取其中使间隔最大的那个理想的超平面。(Fisher 线性判别是寻找法向量，感知机是迭代找到完成分类的超平面，两者都停留在“分”的层面)



回顾：判别函数为 $g(\vec{x}) = w\vec{x} + b$ 时，样本点 x_i 到超平面的几何距离为 $d_i = \frac{|g(\vec{x_i})|}{\|\vec{w}\|}$，其中 $w = (w_1, w_2, \ldots, w_p)$。

对于一个正确分类的点，标签 y_i 与 $g(\vec{x_i})$ 的乘积一定为正值，因此几何距离又可写成：

$$d_i = \frac{y_i g(\vec{x_i})}{\|\vec{w}\|} = \frac{y_i(w\vec{x_i} + b)}{\|\vec{w}\|}$$

上式分子部分又叫样本点的 函数间隔，记为 $\hat{\gamma}_i = y_i(wx_i + b)$，几何距离表示成：$d_i = \frac{\hat{\gamma}_i}{\|w\|}$
两种类别中距离最近的两点到超平面的距离应相同（因为在保证两点到超平面距离和最大的同时，超平面在中间可以保证两类别尽可能地分开），则有：

$$d_{\max} = \frac{\hat{\gamma}}{\|\mathbf{w}\|}, \quad \hat{\gamma} = \min(\gamma_1, \gamma_2, \ldots, \gamma_N)$$

约束条件为：$\hat{\gamma}_i \geq \hat{\gamma}$

可以观察到 $\hat{\gamma}$ 的值是与 w 向量同比例缩放的，那么，我们可以取函数间隔 $\hat{\gamma} = 1$ 来求解，表达式为：

$$d_{\max} = \frac{1}{\|\mathbf{w}\|}$$

约束条件：$\hat{\gamma}_i \geq 1$

转换为优化问题：

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(w \cdot x_i + b) - 1 \geq 0, \quad i = 1, 2, \ldots, N \end{aligned}$$

显然这里需要求解一个含不等式约束条件的最值问题，这里需要利用 拉格朗日对偶性 将原始问题转换成对偶问题，通过解对偶问题得到原始问题的解。以下是对拉格朗日对偶的补充讲解：

约束问题：

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c_i(x) \leq 0, \ i = 1, 2, \ldots, k \\ & h_j(x) = 0, \ j = 1, 2, \ldots, l \end{aligned}$$

拉格朗日函数：

$$L(x, \alpha, \beta) = f(x) + \sum_{i=1}^{k} \alpha_i c_i(x) + \sum_{j=1}^{l} \beta_j h_j(x)$$

α_i, β_j为乘子，$\alpha_i \geq 0$

$$\theta_P(x) = \max_{\alpha, \beta : \alpha_i \geq 0} \left[f(x) + \sum_{i=1}^{k} \alpha_i c_i(x) + \sum_{j=1}^{l} \beta_j h_j(x) \right] = \begin{cases} f(x), & x \text{ 满足原始问题约束} \\ +\infty, & \text{其他} \end{cases}$$

再来考虑极小问题：$\min_x \theta_P(x) = \min_x \max_{\alpha, \beta : \alpha_i \geq 0} L(x, \alpha, \beta)$
原始最优化问题就等价于：$p^* = \min_x \theta_P(x)$
上面我们将带约束的问题等价写成了**原始问题**（广义拉格朗日函数的极小极大问题）：$\min_x \theta_P(x) =$

$\min_x \max_{\alpha,\beta:\alpha_i\geq 0} L(x,\alpha,\beta)$

下一步就是讨论**对偶问题**（广义拉格朗日函数的极大极小问题）：

$$\theta_D(\alpha,\beta)=\min_x L(x,\alpha,\beta)$$

$$\max_{\alpha,\beta:\alpha_i\geq 0}\theta_D(\alpha,\beta)=\max_{\alpha,\beta:\alpha_i\geq 0}\min_x L(x,\alpha,\beta)$$

对偶问题的最优解：

$$d^*=\max_{\alpha,\beta:\alpha_i\geq 0}\theta_D(\alpha,\beta)$$

Q1：什么是对偶问题嘞?

A1：可以观察到，原始问题我们是先对拉格朗日函数在约束范围下的最大，然后再对 x 变量取函数的最小值；而对偶问题的操作刚好相反，是先不管约束，我们对所有范围内的 x 取拉格朗日函数的最小值，再来讨论哪一对约束 α 和 β 可以找到最大的函数值。原始问题和对偶问题的求解顺序是反过来的。

Q2：为啥要有对偶问题嘞?

A2：原始问题的第一步是带约束的优化问题，对于这样一个问题很难解，比如：约束太复杂、变量维度太高。所以我们不直接求原问题的最小，而是找到它的下界，再把下界往上推，推到极限时（$\max_{\alpha,\beta:\alpha_i\geq 0}$ 的过程），就得到原问题的最优值了。

原始问题和对偶问题的关系：如果两者都有最优解，则

$$d^*=\max_{\alpha,\beta:\alpha_i\geq 0}\min_x L(x,\alpha,\beta)\leq\min_x\max_{\alpha,\beta:\alpha_i\geq 0}L(x,\alpha,\beta)=p^*$$

设 x^* 是原始问题的可行解，α^*、β^* 是对偶问题的可行解，且 $d^*=p^*$，那么 x^* 和 α^*、β^* 分别是各自对应问题的最优解。

KKT 条件：KKT 条件是凸优化问题中，最优解必须满足的**充要条件**。假设函数 $f(x)$ 和不等式约束 $c_i(x)$ 都是凸函数，等式约束 $h_j(x)$ 是仿射函数（也就是线性函数）。x^* 和 α^*、β^* 满足 KKT 条件：

$$\begin{aligned}
&\nabla_x L(x^*,\alpha^*,\beta^*)=0, && \alpha_i^* c_i(x^*)=0, && i=1,2,\dots,k\\
&\nabla_\alpha L(x^*,\alpha^*,\beta^*)=0, && c_i(x^*)\leq 0, && i=1,2,\dots,k\\
&\nabla_\beta L(x^*,\alpha^*,\beta^*)=0, && \alpha_i^*\geq 0, && i=1,2,\dots,k\\
& && h_j(x^*)=0, && j=1,2,\dots,l
\end{aligned}$$

根据上述介绍，再来看支持向量机的最优化问题：$L(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^N\alpha_i y_i(w\cdot x_i+b)+\sum_{i=1}^N\alpha_i$

对偶问题为：

$$\max_\alpha\min_{w,b}L(w,b,\alpha)$$

先求 $L(w,b,\alpha)$ 对 w b 的极小值，再求对 α 的极大。

1. 求对 w b 的极小

$$\begin{cases}\nabla_w L(w,b,\alpha) = w - \sum_{i=1}^N \alpha_i y_i x_i = 0 \\ \nabla_b L(w,b,\alpha) = -\sum_{i=1}^N \alpha_i y_i = 0\end{cases} \Rightarrow \begin{cases} w = \sum_{i=1}^N \alpha_i y_i x_i \\ \sum_{i=1}^N \alpha_i y_i = 0\end{cases}$$

$$\begin{aligned}\therefore \min_{w,b} L(w,b,\alpha) &= \frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^N \alpha_i y_i \left(\left(\sum_{j=1}^N \alpha_j y_j x_j\right)\cdot x_i + b\right) + \sum_{i=1}^N \alpha_i \\ &= -\frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^N \alpha_i\end{aligned}$$

2. 求对 α 的极大

$$\begin{aligned}&\max_\alpha \ -\frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^N \alpha_i \\ &\text{s.t.} \quad \sum_{i=1}^N \alpha_i y_i = 0 \\ &\qquad \alpha_i \geq 0,\ i = 1,2,\dots,N\end{aligned} \quad \Rightarrow \quad \begin{aligned}&\min_\alpha \ \frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^N \alpha_i \\ &\text{s.t.} \quad \sum_{i=1}^N \alpha_i y_i = 0 \\ &\qquad \alpha_i \geq 0,\ i = 1,2,\dots,N\end{aligned}$$

3. 设 $\alpha^* = (\alpha_1^*, \alpha_2^*, \dots, \alpha_N^*)^T$ 是对偶最优问题的解，存在下标 j，使得 $\alpha_j^* > 0$;

   对 b^* 的推导：

$$\begin{aligned}&\alpha_j^* \left[y_j\left(w^* \cdot x_j + b^*\right) - 1\right] = 0 \\ \Rightarrow \quad &y_j\left(w^* \cdot x_j + b^*\right) = 1 \quad \xrightarrow{y_j^2 = 1} w^* \cdot x_j + b^* = y_j \\ \Rightarrow \quad &b^* = y_j - \sum_{i=1}^N \alpha_i^* y_i (x_i \cdot x_j)\end{aligned}$$

   则可解得：

$$w^* = \sum_{i=1}^N \alpha_i^* y_i x_i$$

$$b^* = y_j - \sum_{i=1}^N \alpha_i^* y_i (x_i \cdot x_j)$$

   求得分离超平面为：$w^* \cdot x + b^* = 0$

   分类决策函数：$f(x) = \text{sign}(w^* \cdot x + b^*)$

   eg.

正例点 $x_1=(3,3)^T$, $x_2=(4,3)^T$ 负例点：$x_3=(1,1)^T$

对偶形式：$\min_\alpha \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(x_i\cdot x_j)-\sum_{i=1}^{N}\alpha_i=\frac{1}{2}(18\alpha_1^2+25\alpha_2^2+2\alpha_3^2+42\alpha_1\alpha_2$

$-12\alpha_1\alpha_3-14\alpha_2\alpha_3)-\alpha_1-\alpha_2-\alpha_3$

s.t. $\alpha_1+\alpha_2-\alpha_3=0$ $\alpha_i\geq 0$, $i=1,2,3$

代入得目标函数：$s(\alpha_1,\alpha_2)=4\alpha_1^2+\frac{13}{2}\alpha_2^2+10\alpha_1\alpha_2-2\alpha_1-2\alpha_2$

$\begin{cases}\frac{\partial s(\alpha_1,\alpha_2)}{\partial\alpha_1}=8\alpha_1+10\alpha_2-2=0\\ \frac{\partial s(\alpha_1,\alpha_2)}{\partial\alpha_2}=13\alpha_2+10\alpha_1-2=0\end{cases}\Rightarrow\begin{cases}\alpha_1=\frac{3}{2}\\ \alpha_2=-1\end{cases}$ 不满足约束 $\alpha_i\geq 0$

∴最小值在边界取得：①$\alpha_1=0$，最小 $s(0,\frac{2}{13})=-\frac{2}{13}$

②$\alpha_3=0$，最小 $s(\frac{1}{4},0)=-\frac{1}{4}$ $\Rightarrow\begin{cases}\alpha_1=\frac{1}{4}\\ \alpha_2=0\\ \alpha_3=\frac{1}{4}\end{cases}$

∴$\alpha_1^*=\alpha_3^*=\frac{1}{4}$，$w^*=(\frac{1}{2},\frac{1}{2})$，$b^*=-2$ $\Rightarrow$ 超平面：$\frac{1}{2}x^{(1)}+\frac{1}{2}x^{(2)}-2=0$

5.2 线性不可分支持向量机

现实数据中常常含有噪声、异常点和类间重叠，”完全线性可分”通常不成立。硬间隔模型要求所有样本都严格满足间隔约束，容易被少数离群点破坏，导致决策边界不合理。

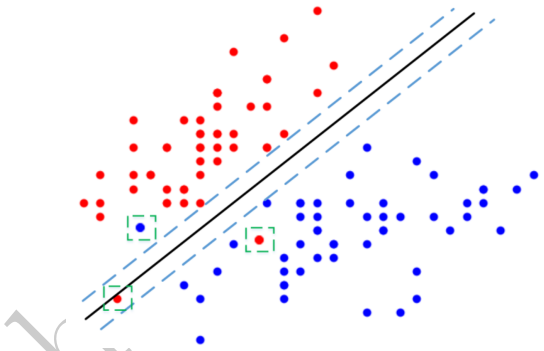


图 13: 线性不可分

对于这种存在错误的点被分类或在间隔里，引入亏损 ξ_i，也称为松弛变量，由此约束问题变为：

$$\begin{aligned}\min_{w,b,\xi}\quad & \frac{1}{2}\|w\|^2+C\sum_{i=1}^{N}\xi_i\\ \text{s.t.}\quad & y_i(w\cdot x_i+b)\geq 1-\xi_i,\quad i=1,2,\ldots,N\\ & \xi_i\geq 0,\quad i=1,2,\ldots,N\end{aligned}$$

原始问题的拉格朗日函数：

$$L(w,b,\xi,\alpha,\mu)=\frac{1}{2}\|w\|^2+C\sum_{i=1}^{N}\xi_i-\sum_{i=1}^{N}\alpha_i\left(y_i(w\cdot x_i+b)-1+\xi_i\right)-\sum_{i=1}^{N}\mu_i\xi_i$$

其中：$\alpha_i\geq 0,\ \mu_i\geq 0$

求解过程：

1. 求 $L(w,b,\xi,\alpha,\mu)$ 对 w,b,ξ 的极小：

$$\begin{cases}\nabla_w L(w,b,\xi,\alpha,\mu)=w-\sum_{i=1}^N \alpha_i y_i x_i=0\\ \nabla_b L(w,b,\xi,\alpha,\mu)=-\sum_{i=1}^N \alpha_i y_i=0\\ \nabla_\xi L(w,b,\xi,\alpha,\mu)=C-\alpha_i-\mu_i=0\end{cases}\Rightarrow\begin{cases}w=\sum_{i=1}^N \alpha_i y_i x_i\\ \sum_{i=1}^N \alpha_i y_i=0\\ C=\alpha_i+\mu_i\end{cases}$$

$$\begin{aligned}\min_{w,b,\xi} L(w,b,\xi,\alpha,\mu)=&\frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_j y_i y_j(x_i\cdot x_j)+(\alpha_i+\mu_i)\sum_{i=1}^N \xi_i-\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_j y_i y_j(x_i\cdot x_j)\\ &+\sum_{i=1}^N \alpha_i-\sum_{i=1}^N \alpha_i\xi_i-\sum_{i=1}^N \mu_i\xi_i\\ =&-\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^N \alpha_i\alpha_j y_i y_j(x_i\cdot x_j)+\sum_{i=1}^N \alpha_i\end{aligned}$$

2. 再对 $\min_{w,b,\xi} L(w,b,\xi,\alpha,\mu)$ 求 α 的极大，得到对偶问题：

$$\begin{aligned}\max_\alpha\ &-\frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_j y_i y_j(x_i\cdot x_j)+\sum_{i=1}^N \alpha_i\\ \text{s.t.}\ &\sum_{i=1}^N \alpha_i y_i=0\\ &C-\alpha_i-\mu_i=0\\ &\alpha_i\ge 0\\ &\mu_i\ge 0,\ i=1,2,\dots,N\end{aligned}\quad\Rightarrow\quad\begin{aligned}\min_\alpha\ &\frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_j y_i y_j(x_i\cdot x_j)-\sum_{i=1}^N \alpha_i\\ \text{s.t.}\ &\sum_{i=1}^N \alpha_i y_i=0\\ &0\le\alpha_i\le C,\ i=1,2,\dots,N\end{aligned}$$

3. 设 $\alpha^*=(\alpha_1^*,\alpha_2^*,\dots,\alpha_N^*)^T$ 是对偶最优问题的解，存在下标 j，使得 $0<\alpha_j^*<C$;

   得到原始问题的解 w^*,b^*：

$$w^*=\sum_{i=1}^N \alpha_i^* y_i x_i$$

$$b^*=y_j-\sum_{i=1}^N \alpha_i^* y_i(x_i\cdot x_j)$$

求得分离超平面为：$w^*\cdot x+b^*=0$

分类决策函数：$f(x)=\text{sign}(w^*\cdot x+b^*)$

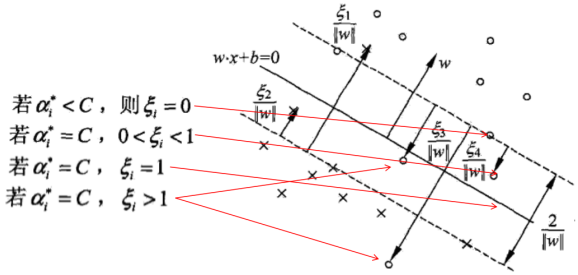
可以观察到，软间隔 SVM 和硬间隔 SVM 的最优解表达式居然是一样的，不同之处 α^* 的范围，硬间隔 SVM 为 $\alpha_i^*\ge 0$，软间隔 SVM 为 $0\le\alpha_i\le C$

Q：这里面的 C 啥意思呢？又有什么用呢？

A：C 是惩罚系数，用来控制**模型复杂度**与 **训练误差**之间的权衡。如下是不同 C 与 ξ 所决定的超平面图。软间隔 SVM 目标函数：$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i$ 中，前一项控制间隔大小，后一项控制训练误差。

当 C 很大时，目标函数会被后一项主导，为了取得 min 则会使 ξ 都被压缩到很小，导致为了拟合所有样本，模型的决策边界间隔被压得很**窄**，学到了大量噪声和异常点的特征，泛化能力差，易**过拟合**。

当 C 很小时，目标函数会被前一项主导，模型会优先扩大间隔，而不太在意训练误差。少量样本的误分类或进入间隔带来的惩罚很小，模型会主动牺牲一些训练精度来换取更宽的间隔。模型更简单，但有**欠拟合**的风险。



合页损失函数：样本点的函数间隔有 $\hat{\gamma}_i = y_i(wx_i + b)$，其在一定程度上反映了到超平面的距离，在线性可分的情况中就已经明确最近距离等比缩放为 1，因此引入合页损失函数，对间隔中的点施加惩罚。

$$L(z) = [1 - z]_+$$

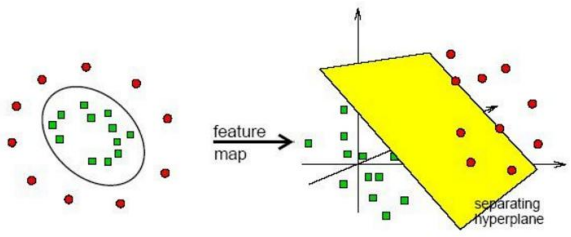
其中：$[u]_+ = \max(u, 0)$；当样本被正确分类且距离边界足够远时，损失为 0；当样本落入间隔内部或被误分类时，损失为正。

软间隔 SVM 则等价于优化问题：

$$\min_{w,b} \sum_{i=1}^{N} [1 - y_i (w \cdot x_i + b)]_+ + \lambda\|w\|^2$$

5.3 核函数

许多现实中的数据在原始的空间中是线性不可分的，但是在更高维特征空间中可能线性可分，如下图所示。特征映射：$x \mapsto \phi(x)$。



当然我们不能直接“暴力升维”，否则会导致计算快速上升、样本稀疏难统计，且容易**过拟合**。选取

核函数的技巧，只在对偶问题中使用内积，将 $x_i \cdot x_j$ 替换为核函数：

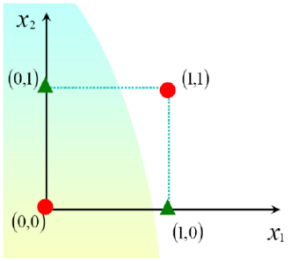
$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

- 多项式核：$K(x_i, x_j) = (x_i \cdot x_j + 1)^p$ 适用于多项式特征，p 为多项式次数。
- 高斯核：$K(x, z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$ 适用于复杂非线性，σ 控制宽度。
- 字符串核：$K(s, t) = \sum_i \phi_i(s)\phi_i(t)$ 适用于文本数据，衡量字符串相似度。

核 SVM 决策函数：

$$f(x) = \text{sign}\left(\sum_i \alpha_i^* y_i K(x_i, x) + b^*\right)$$

eg.



$\omega_1 : \{(0,0),(1,1)\}$
$\omega_2 : \{(1,0),(0,1)\}$

线性识别函数不存在
把2维空间(x_1, x_2)
变为3维空间(x_1, x_2, x_3)

$x_3 = x_1 x_2$

(x_1, x_2)	(x_1, x_2, x_3)
(0,0)	(0,0,0)
(1,1)	(1,1,1)
(1,0)	(1,0,0)
(0,1)	(0,1,0)

6 k-近邻和统计学习理论

6.1 k-近邻

1. 参数学习：先确定学习机器实现的函数集，然后从函数集中选择最优函数，通过训练数据估计参数。如：线性回归、逻辑回归、SVM、神经网络。
2. 通过对训练样本的学习直接构建分类机器，不预设函数形式，其无法用一个包含若干待定参数的函数来表示。如：k 近邻（KNN）、核密度估计、高斯过程。

k-近邻算法（k-NN）是一种简单且直观的监督学习算法，用于分类和回归任务。其基本思想是：给定一个待分类样本，找出与其距离**最近的 k 个训练样本**，然后通过这 k 个邻居的类别来决定待分类样本的类别。(这依旧是一个分类问题，但是不同于感知机和 SVM 通过超平面将样本进行分类，该方法无需函数的形式，而是通过最邻近的 k 个样本的类别来预测)

k-近邻法模型的三个基本要素：**距离度量**、**k 值**的选择和**分类决策规则**。

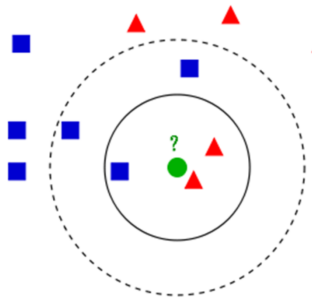


图 14: k-近邻法示意图

6.1.1 距离度量

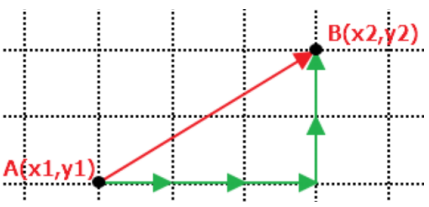
特征空间中两个实例点的距离是两个实例点**相似程度**的反映。因此 k-近邻法的重要问题就是计算样本之间的距离，有 Lp/Minkowski 距离：

$$d(A,B)=\left(\sum_{i=1}^{n}|x_{Ai}-x_{Bi}|^{p}\right)^{\frac{1}{p}}$$

p=1 时为曼哈顿距离（L1 距离）：$d(A,B)=\sum_{i=1}^{n}|x_{Ai}-x_{Bi}|$

p=2 时为欧式距离（L2 距离）：$d(A,B)=\sqrt{\sum_{i=1}^{n}(x_{Ai}-x_{Bi})^{2}}$

随着 p 值增大，距离度量对大差异维度更加敏感，不同的距离度量会导致不同的最近邻选择，进而影响分类结果。如下图可以展示 L1 距离和 L2 距离是什么样的：



6.1.2 k 值选取

k 值的选择会对 k 近邻法的结果产生重大影响。k 值决定了模型的复杂度和泛化能力。**近似误差**指训练集上的误差（可以理解为偏差 Bias），**估计误差**指测试集上的误差（可以理解为方差 Variance）。

- k 较小时，相当于仅考虑较小范围内的训练实例进行预测，近似误差减小，但是考虑的点较少，预测结果对近邻的实例点非常敏感，容易受到噪声影响，估计误差大。决策边界会变得极其细碎、高度曲折，只要有一个点变了边界都会变化，因此**模型更复杂，易过拟合**。
- k 较大时，考虑范围内的实例更多，近似误差增大，但是对单个点/噪声的变化不敏感，估计误差减小，预测结果更加稳定，**模型也更加简单，但可能欠拟合**。

在应用中，k 值一般取一个比较小的数值。通常采用交叉验证法来选取最优的 k 值。

6.1.3 分类决策规则

k 近邻法中的分类决策规则往往是**多数表决**，即由输入实例的 k 个邻近的训练实例中的多数类决定输入实例的类。如果分类的损失函数为 0-1 损失函数，预测正确则损失为 0，预测错误则损失为 1，误分类率有：$\frac{1}{k}\sum I(y_i \neq c_j)$，$I()$ 代表 0-1 损失值，最终选取结果为误分类率最小。

6.1.4 Kd 树

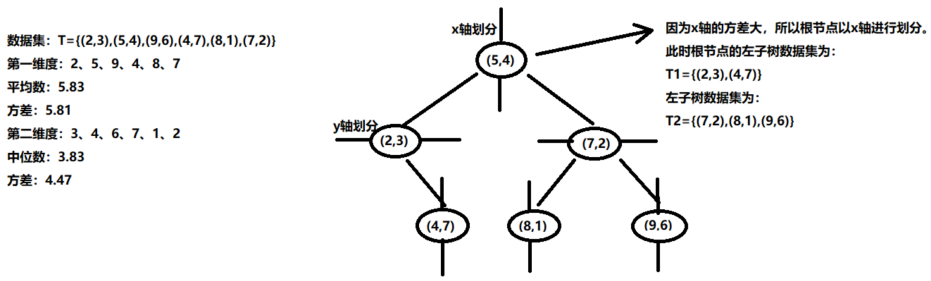
实现 k 近邻时核心是对训练数据进行快速 k 近邻搜索，最简单的实现是线性扫描（穷举搜索），即计算输入实例与每一个训练实例的距离，计算并存储好以后，再查找 K 近邻。但是当训练集很大时，计算十分耗时，时间复杂度为 O(n)。因此考虑使用特殊的结构存储训练数据，以减小计算距离的次数。

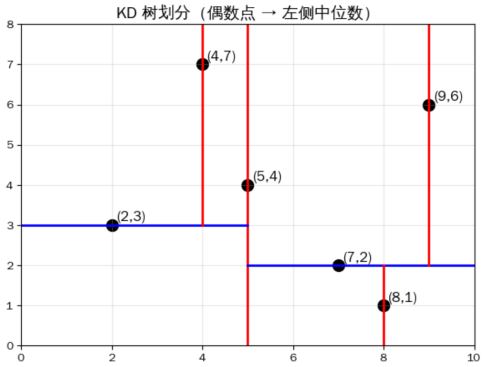
Kd 树的思路则是把高维空间的数据划分成小区域，寻找近邻时只需要在相关小区域进行搜索，平均时间复杂度降为 O(log n)。Kd 树是二叉树，表示对 k 维空间的一个划分。

构建方法：

（1）**构造根结点**——使根结点对应于 k 维空间中包含所有实例点的**超矩形区域**（可以理解为对应的能装下所有训练数据的“多维盒子”）；
（2）**递归切分**——选择坐标轴和切分点，用垂直于坐标轴的超平面切分空间；
（3）**生成子结点**——将实例分到左、右两个子区域，直到子区域没有实例。

为平衡 Kd 树，一般依次选取坐标轴对空间切分，且选择训练实例点在选定坐标轴上的中位数为切分点。对于偶数个点则选取靠左/靠右的中位数，切分可以默认先 x 维切分，也可以用方差最大维度切分（最优）。如下是一个构建 Kd 树实例：





构建好 Kd 树后就是对测试点进行最近邻域搜索，以测试点（2.1，3.1）为例

（1）**从根结点向下搜索**：根据 Kd 树划分进入左子树，再按照 y 轴划分进入右子树为 (4,7)，当前节点为叶子节点，无法继续向下搜索。此时可以将当前节点 (4,7) 作为候选最近邻点。
（2）**回溯与剪枝**：回溯到父节点 (2,3) 发现测试点到该节点距离更近，将当前节点 (2,3) 作为候选最近邻点。再回溯到根节点 (5,4) 比较比较，最终确认最近邻点为 (2, 3)，而右枝可以直接剪掉不考虑。

6.2 统计学习理论

传统统计学是建立在大样本、渐近收敛的基础上，但是现实任务中往往面临有限个小样本的场景，传统统计的大数定律、渐近理论不再适用，因此有了统计学习理论。

在第一章就已经提及，我们依据**期望风险（真实风险）**：$R_{exp}(f) = E_P[L(Y, f(X))] = \int_{x*y} L(y, f(x))P(x, y)dxdy$ 来衡量模型预测的好坏，但由于联合分布未知，因此选取**经验风险**：$R_{emp}(f) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i))$ 近似地衡量。统计学理论把这种在训练样本上最小化错误或者风险的策略称为**经验风险最小化（ERM）原则**。然而，当样本数量有限时，最小化经验风险并不能保证期望风险也最小，还会有过拟合的风险。因此，统计学习理论需要解决的**核心问题**：（1）学习过程的一致性——在什么条件下，经验风险最小化能收敛到期望风险最小；（2）收敛速度——经验风险收敛到期望风险的速度有多快，如何量化；（3）控制推广能力——如何控制学习机器的推广能力，使其在小样本情况下也能表现良好。

- 欠学习（欠拟合）：模型**过于简单**，无法捕捉数据中的潜在规律，导致在训练集和测试集上都表现不佳的现象。**训练误差和测试误差都偏高，偏差高、方差低**。可能是模型选择过简单、特征选择不当、训练不充分、正则化中惩罚过大。
- 过学习（过拟合）：指模型过于复杂，不仅学习了数据的真实规律，还学习了训练数据中的噪声和随机波动，导致在训练集上表现很好但在测试集上表现差。**训练误差很低，测试误差偏高，偏差低、方差高**。可能是模型复杂度过高、训练数据不足、训练时间过长、噪声干扰导致。

- 学习过程的一致性：指在训练样本上以经验风险最小化原则进行的学习，在样本数趋近于无穷大时与期望风险最小的目标是否一致。ERM 原则一致性的充要条件就是**经验风险在整个函数集上一致收敛于期望风险**
- 收敛速度：描述学习过程达到最佳性能的速度——$\mathbb{E}\left[R(f_N) - R(f^*)\right] = O\left(N^{-\alpha}\right)$，$R(f_N)$ 是模型

在 N 个样本上的训练误差；$R(f^*)$ 是真实误差；$O\left(N^{-\alpha}\right)$ 表示这个差距会随着样本数 N 的增加，以 $N^{-\alpha}$ 的速度缩小，通常 $\alpha \in [0.5, 1]$，越大收敛速度越快。

- 样本复杂度:指达到指定精度所需的样本量,对目标精度 ε 和置信度 $1-\delta$ 有:$P\left(R(f_N) - R(f^*) \leq \varepsilon\right) \geq 1-\delta$；找到最小的样本复杂度 N。

- 赤池信息量准则（AIC）：是经典的模型选择工具，在**模型拟合度**和**模型复杂度**之间找平衡，目标是寻找到最好解释数据且参数最少的模型。公式为 $AIC = 2k - 2ln(L)$；k——参数的数量；L——最大似然函数值（代表模型对数据的拟合程度）。AIC 值越小，模型越好；优先选择 AIC 值最小的模型，如果多个模型 AIC 值相近，选择更简单的模型。

  当样本量 n 较小时，AIC 需要修正为 **AICc**（因为小样本 AIC 倾向于选择过拟合的复杂模型）：$\text{AICc} = \text{AIC} + \frac{2k(k+1)}{n-k-1}$，当 n 趋于无穷，AICc 收敛于 AIC。

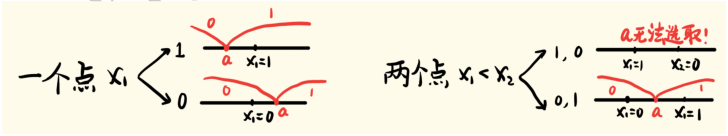
- 贝叶斯信息准则（BIC）：同样是用来在模型拟合度和复杂度之间做平衡的模型选择工具，公式为 $BIC = k\ ln(n) - 2ln(L)$；n 为样本容量（训练数据的数量）；可见 BIC 对参数的惩罚是与样本容量相关的，n>7 时，BIC 的惩罚更强。

  小样本时 AIC 倾向选择更复杂模型；大样本时 BIC 倾向选择更简单模型。AIC 的设计目标就是最小化预测误差，因此**以预测为主时选取 AIC**；BIC 对模型复杂度的惩罚更重，会强制模型向更**简洁**的方向收敛，优先选择参数更少、结构更简单的模型，适用于**解释**任务。

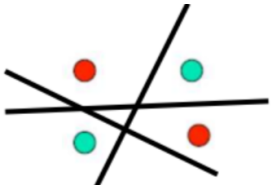
- VC 维：指**能够被假设空间完全打散的最大样本点数**，记作 $VC(H)$，其中 H 是假设空间。用于定量衡量模型的复杂度和表达能力。

  打散——对于 n 个点的所有 2^n 种标签组合，假设空间中都存在一个假设能够正确分类。

  示例：1）**一维阈值函数（VC (H)=1）**：公式为 $H = \{f(x) = I(x \geq a) \mid a \in \mathbb{R}\}$，在数轴上找一个阈值 a，把右边判为 “1”、左边判为 “0”。最多可打散一个点，VC (H)=1 可结合下图理解：



  2）**二维线性分类器（VC (H)=3）**：公式为 $H = \left\{f(x) = \text{sign}(wx + b) \mid w \in \mathbb{R}^2, b \in \mathbb{R}\right\}$，是二维平面上的直线分类模型。如下图可知 4 个点无法完全打散，因此 VC(H)=3。



  3）**n 维线性超平面（VC (H)=n+1）**：公式为 $H = \{f(x) = \text{sign}(wx + b) \mid w \in \mathbb{R}^n, b \in \mathbb{R}\}$

  4）**二维轴对称矩形（VC (H)=4）**：用和坐标轴对齐的矩形框，把点框起来判为正。4 个点有 16 中标注方式，把 4 个点放在矩形四个角落时，每种标注都能找到对应的矩形。

函数集的容量——指函数集能够实现的分类能力，即函数集的表达能力或复杂度。**VC 维越大，模型容量越大，函数集能够表达的分类边界越复杂**。

***VC 维与推广能力**——经验风险与期望风险的差距可以用 VC 维来界定，这个界被称为**推广能力的界**。有公式：

$$R(\alpha) \leq R_{\text{emp}}(\alpha) + \sqrt{\frac{h\left(\ln\frac{2l}{h}+1\right) - \ln\frac{\eta}{4}}{l}}$$

其中 $R(\alpha)$ 为模型的真实误差；R_{emp} 为经验风险；h 为 VC 维；l 为样本数；η 为置信度参数，$1-\eta$ 是置信水平。

可见 **VC 维越大**，置信范围越大，经验风险与期望风险差距越大，**过拟合风险越高**，样本需求多。**样本数越多**，置信范围越小，泛化能力更强。因此，控制 VC 维是控制模型复杂度、避免过学习的关键。

- 结构风险最小化（SRM）原则：引入模型复杂度惩罚项，最小化训练误差的同时控制复杂度，避免过学习。结构风险定义公式：

$$R_{\text{srm}}(\alpha) = R_{\text{emp}}(\alpha) + \Phi(h/l)$$

$R_{\text{emp}}(\alpha)$ 为经验风险；$\Phi(h/l)$ 为置信范围（复杂度惩罚项）

实现 SRM 有两种策略：一种是**固定置信范围**，在函数集中选取使经验风险最小的函数（常用方法，通过交叉验证选取最优模型）；另一种方法是**固定经验风险**，在函数集中选取使得置信范围最小的函数。

- **正则化方法**在前面章节已经对此有过基本了解了。其通过在目标函数中添加模型复杂度惩罚项，在最小化训练误差的同时，控制模型复杂度，从而实现结构风险最小化。通用形式：

$$\min : R_{\text{emp}}(\alpha) + \lambda \cdot \Omega(\alpha)$$

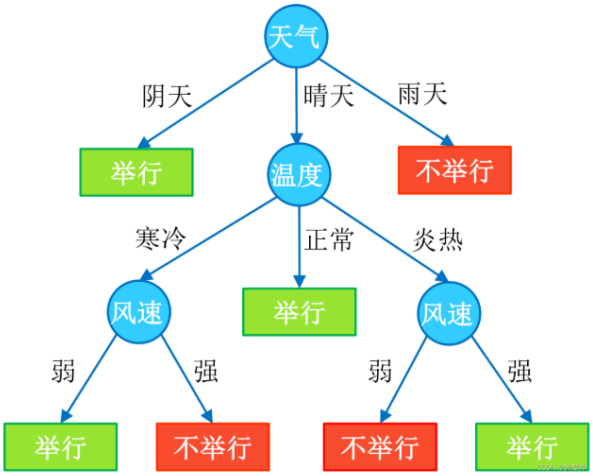
λ 为正则化系数，控制惩罚强度；$\Omega(\alpha)$ 为惩罚项，衡量模型复杂度。

正则化可以防止过拟合，提高泛化能力，处理不适定问题使优化问题有唯一解。L1 正则化（LASSO），L2 正则化（Ridge），具体可见第二章线性回归。

7 决策树与集成学习

7.1 决策树

决策树是一种以树形结果展示决策规则和分类结果的模型，属于非参学习方法，本质是一种符合生活直观的归纳分类方法。如下图示例（是否举行某个活动）可以展示决策树的运作机制，基于此决策树就可以根据已有的相关信息对是否举行活动进行预测。



7.1.1 组成

决策树的基本组成部分为：决策结点、分支、叶子

- 根结点：整个决策树的开始（如上图“天气”为根结点）
- 叶结点：代表一种可能的分类结果（如上图“举行”/“不举行”的判断）
- 每个分支是一个新的决策结点或者叶子
- 每个决策结点代表一个问题或者对应待分类对象的属性

7.1.2 ID3 方法

ID3 是决策树学习方法中最具代表性的经典算法，核心是用**信息增益**作为选择分支，构建决策树。在此需要引入以下部分概念。

- 信息量：用于衡量事件“意外程度”的指标，事件越不确定、越稀有，信息量就越大。对于某一事件 x，发生概率为 P(x)，信息量为：

$$I(x) = -\log_2 P(x)$$

- 熵：用于衡量系统不确定性的指标，是随机变量多种可能的平均信息量，熵越大，随机变量不确定性越大。随机变量 Y 有概率分布：$P(Y = y_i) = p_i,\ i = 1, 2, \ldots, n$，随机变量 Y 的熵定

义为：$H(Y) = -\sum_{i=1}^{n} p_i \log_2 p_i$，可见熵仅与 Y 的分布有关，与 Y 取值无关。有范围

$$0 \le H(Y) \le \log n \quad \text{n 为随机变量 Y 可能取值个数}$$

当 Y 为 0-1 分布时，$H(Y) = -p\log_2 p - (1-p)\log_2(1-p)$

- 条件熵：在已知随机变量 X 的条件下随机变量 Y 的不确定性。

$$H(Y|X) = \sum_{i=1}^{n} p_i H(Y|X = x_i)$$

注意这里的概率 p_i 指的是特征 $X = x_i$ 的概率。

- **信息增益**：得知特征 X 的信息而使得类 Y 的信息的不确定性减少的程度，也称为互信息，表示为 $H(Y) - H(Y|X)$

选取 ID3 算法进行决策树构建时，先从根结点开始，对结点计算所有可能的特征的信息增益，选择**信息增益最大**的特征作为结点的特征，由该特征的不同取值建立子结点；再对子结点递归地调用以上方法，构建决策树，直到所有特征的信息增益均很小或没有特征可以选择为止；最终得到一棵决策树。

7.1.3 示例分析

选取如下是否购买计算机的例子，进行决策树 ID3 算法的构建。

计数	年龄	收入	学生	信誉	归类：买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

1. 计算决策属性（Y）熵：决策属性为“买” / “不买”

属性	人数	概率	熵
Y_1（买）	641	P_1=0.62660	$H(Y) = -P_1\log_2 P_1 - P_2\log_2 P_2 = 0.95537$
Y_2（不买）	383	P_2=0.37440	

2. 计算各个条件属性熵

1）计算年龄（A）的熵和信息增益

年龄 (A)	购买情况	概率	熵 $H(Y\|A_i)$	人数占比
A_1 (青年)	128 买 256 不买	$P_1(\text{买})=\frac{128}{384}$ $P_2(\text{不买})=\frac{256}{384}$	$H(Y\|A_1)=-P_1\log_2 P_1-P_2\log_2 P_2=0.9183$	$\frac{384}{1024}$
A_2 (中年)	256 买 0 不买	$P_1(\text{买})=\frac{256}{256}$ $P_2(\text{不买})=\frac{0}{256}$	$H(Y\|A_2)=-P_1\log_2 P_1-P_2\log_2 P_2=0$	$\frac{256}{1024}$
A_3 (老年)	257 买 127 不买	$P_1(\text{买})=\frac{257}{384}$ $P_2(\text{不买})=\frac{127}{384}$	$H(Y\|A_3)=-P_1\log_2 P_1-P_2\log_2 P_2=0.9157$	$\frac{384}{1024}$
条件熵 $H(Y\|A)=\frac{384}{1024}\times 0.9183+\frac{256}{1024}\times 0+\frac{384}{1024}\times 0.9157=0.6877$				
年龄的信息增益 $G(A)=H(Y)-H(Y\|A)=0.9537-0.6877=0.2660$				

2）计算收入（B）的熵和信息增益

收入 (B)	购买情况	概率	熵 $H(Y\|B_i)$	人数占比
B_1 (高)	160 买 128 不买	$P_1(\text{买})=\frac{160}{288}$ $P_2(\text{不买})=\frac{128}{288}$	$H(Y\|B_1)=-P_1\log_2 P_1-P_2\log_2 P_2=0.9911$	$\frac{288}{1024}$
B_2 (中)	289 买 191 不买	$P_1(\text{买})=\frac{289}{480}$ $P_2(\text{不买})=\frac{191}{480}$	$H(Y\|B_2)=-P_1\log_2 P_1-P_2\log_2 P_2=0.9697$	$\frac{480}{1024}$
B_3 (低)	192 买 64 不买	$P_1(\text{买})=\frac{192}{256}$ $P_2(\text{不买})=\frac{64}{256}$	$H(Y\|B_3)=-P_1\log_2 P_1-P_2\log_2 P_2=0.8113$	$\frac{256}{1024}$
条件熵 $H(Y\|B)=\frac{288}{1024}\times 0.9911+\frac{480}{1024}\times 0.9697+\frac{256}{1024}\times 0.8113=0.9361$				
收入的信息增益 $G(B)=H(Y)-H(Y\|B)=0.9537-0.9361=0.0176$				

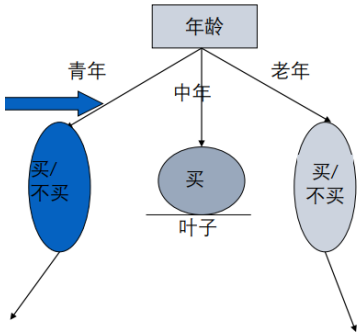
3）计算学生（C）的熵和信息增益

学生 (C)	购买情况	概率	熵 $H(Y\|C_i)$	人数占比
C_1 (学生)	420 买 64 不买	$P_1(\text{买})=\frac{420}{484}$ $P_2(\text{不买})=\frac{64}{484}$	$H(Y\|C_1)=-P_1\log_2 P_1-P_2\log_2 P_2=0.5635$	$\frac{484}{1024}$
C_2 (非学生)	221 买 319 不买	$P_1(\text{买})=\frac{221}{540}$ $P_2(\text{不买})=\frac{319}{540}$	$H(Y\|C_2)=-P_1\log_2 P_1-P_2\log_2 P_2=0.9761$	$\frac{540}{1024}$
条件熵 $H(Y\|C)=\frac{484}{1024}\times 0.5635+\frac{540}{1024}\times 0.9761=0.7811$				
学生的信息增益 $G(C)=H(Y)-H(Y\|C)=0.9537-0.7811=0.1726$				

4）计算信誉（D）的熵和信息增益

信誉 (D)	购买情况	概率	熵 $H(Y\|D_i)$	人数占比
D_1 (良好)	480 买 192 不买	$P_1(\text{买})=\frac{480}{672}$ $P_2(\text{不买})=\frac{192}{672}$	$H(Y\|D_1)=-P_1\log_2 P_1-P_2\log_2 P_2=0.8631$	$\frac{672}{1024}$
D_2 (优秀)	161 买 191 不买	$P_1(\text{买})=\frac{161}{352}$ $P_2(\text{不买})=\frac{191}{352}$	$H(Y\|D_2)=-P_1\log_2 P_1-P_2\log_2 P_2=0.9923$	$\frac{352}{1024}$
条件熵 $H(Y\|D)=\frac{672}{1024}\times 0.8631+\frac{352}{1024}\times 0.9923=0.9075$				
信誉的信息增益 $G(D)=H(Y)-H(Y\|D)=0.9537-0.9075=0.0462$				

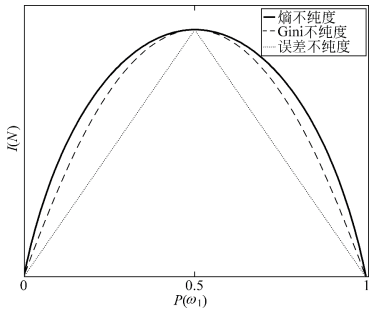
3. 比较信息增益，选取根结点。年龄的信息增益最大，因此以年龄作为根结点，可以划分出**青年**、**中年**、**老年**三支，中年由于全是“买”，因此直接到叶子。而对于青年和老年则需要进一步在剩下三个指标——**收入**、**学生**、**信誉**分别再来计算各自的信息增益，最后一步一步将决策树完成。



7.1.4 其他决策树方法

- 其他不纯度度量：1）Gini 不纯度（方差不纯度）（CART 算法用）——即随机抽取两个样本，其类别不一样的概率，有 $H(Y)=\sum_{m\neq n}P(\omega_m)P(\omega_n)=1-\sum_{j=1}^{k}P^2(\omega_j)$；

  2）误差不纯度——公式为 $H(Y)=1-\max_j P(\omega_j)$，节点里最多的类别占比越高，节点越“纯”。



- C4.5 算法：当一个特征的取值越多，每个取值对应的子集数越少时，条件熵也就越小，信息增益越大。**极端一点想**，比如在讨论一个人买不买计算机时，我们如果加入特征——身份证号，每个样本的身份证号都是不一样的，导致有很多分支，每个分支含有 1 个样本，得到的条件熵很小，信息增益很大，那么我们建决策树会以身份证号为根结点，而非先考虑年龄、性别之类的，显然这是不符合常理的。

这反映了 **ID3 算法在进行进行选取时，容易偏向取值多的特征**，导致模型**过拟合**，缺乏泛化能力。为了修正信息增益带来的偏向性，**C4.5 算法引入了信息增益率**作为标准，有：

$$\Delta G = \frac{H(Y) - H(Y|X)}{H(X)}$$

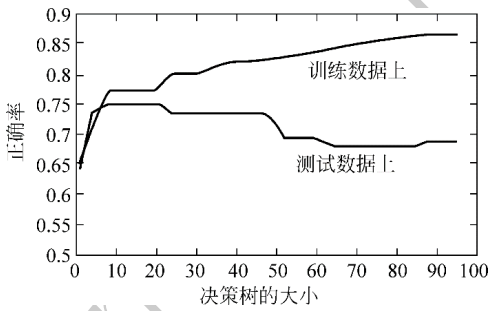
可见，当一个特征取值越多时，其固有信息 $H(X)$ 也会增大，信息增益率也越小，起到了对多取值特征的“惩罚”作用。

- 分类回归树 CART 算法：每个节点上都采用二分法，最后构成二叉树。

7.1.5 决策树的剪枝

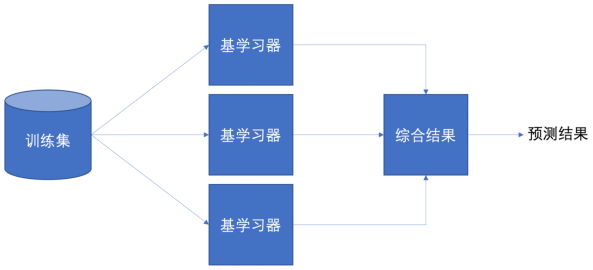
合适的决策树我们希望是**规模小但是在测试集的准确率高**，而无论何种建树方法都会有过拟合、过度分裂、树高过高等现象，如下图展示训练集和测试集的正确率。因此决策树的构建需要人为限制和干预，即**剪枝**可以用于处理过拟合。

剪枝方法有**先剪枝**（阈值法、信息增益的统计显著性分析、交叉验证）和**后剪枝**（最小代价与复杂性的折衷、交叉验证）



7.2 集成学习

集成学习是一种经典的机器学习范式，通过结合多个**单模型**（弱学习器/基学习器）来弥补单个模型的不足，得到一个更加强大和准确的学习系统。



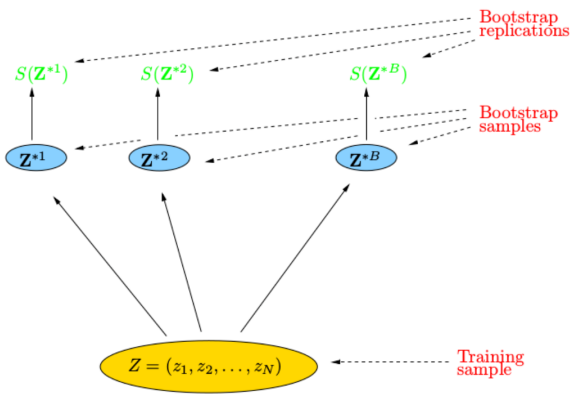
- 强可学习：一个概念（类）存在一个多项式的学习算法能够学习它，并且正确率很高
- 弱可学习：一个概念（类）存在一个多项式的学习算法能够学习它，学习的正确率仅比随机猜略好
- 一个概念是强可学习的充分必要条件是这个概念是弱可学习。因此只要找到一个比随机猜测略好的弱学习算法就可以直接将其提升为强学习算法，而不必直接去找很难获得的强学习算法

- 获得不同的弱分类器有多种方式：使用**不同的弱学习算法**得到不同基本学习器——参数估计、非参数估计；使用**相同的弱学习算法**，但用**不同的参数**——K-Mean 不同的 K，神经网络不同的隐含层；相同输入对象的**不同表示**凸显事物不同的特征；使用不同的训练集——bagging,boosting

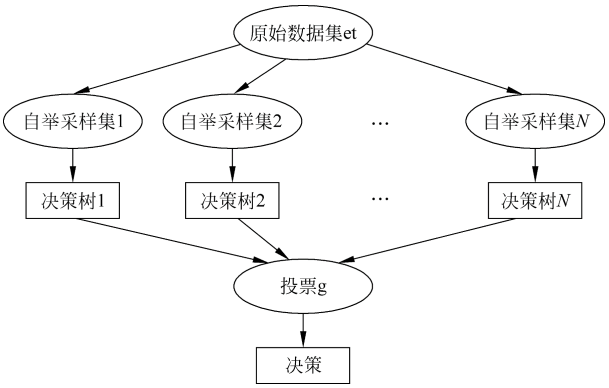
7.2.1 Bagging

Bagging（Bootstrap Aggregating）（自举汇聚法）是集成学习中最经典的并行集成范式，也是随机森林的基础。其核心流程如下：

1. 从原始数据集 $Z=(z_1,z_2,\ldots,z_N)$ 中有放回地抽取一个样本，抽取 N 次，生成一个和原数据集大小相同的新数据集 Z^{*1}，重复这个过程 B 次，得到 B 个独立的训练集 $Z^{*1},Z^{*2},\ldots,Z^{*B}$
2. 并行训练基学习器，用同一个学习算法分别在这 B 个数据集上训练，得到 B 个独立的分类器 $S(Z^{*1}),S(Z^{*2}),\ldots,S(Z^{*B})$
3. 组合预测结果，采取多数投票法，哪个类别被最多分类器选中，就作为最终预测结果。



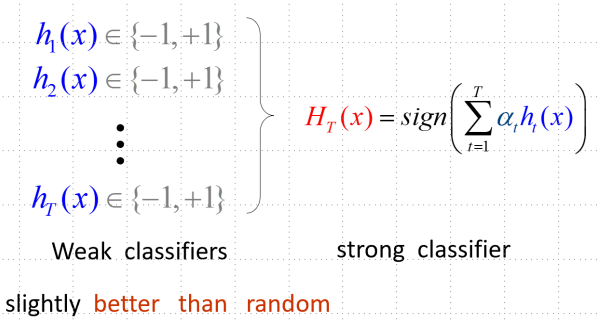
随机森林（Random Forest）算法是 Bagging 的改进，专门针对决策树设计。其流程第一步与 Bagging 一样，得到多个随机抽取的样本集；第二步则是用每个样本集作为训练样本构造决策树，但是在构造决策树过程中，每次是从所有候选特征中**随机抽取 m 个特征**，作为当前节点下决策的备选特征；最后对决策树进行投票，得票最多的类作为随机森林的决策。



随机森林在建树的过程每次是随机选取 m 个特征，这样可以避免所有树都用全部特征而优先学习到最强特征/噪声，导致过拟合出现。

7.2.2 Boosting

Boosting 是一类串行式集成学习框架，其核心是将一组弱可学习器通过迭代加权组合，构造一个强可学习器，每一轮迭代依赖前一轮的学习结果进行调整。（可见与 Bagging 不同，Bagging 可以理解为并行的多轮训练；Boosting 是“串联”地训练）



AdaBoost（Adaptive Boosting）是一种经典的 Boosting 算法。其通过提高那些被前一轮弱分类器错误分类样本的权值，降低那些被正确分类样本的权值来。具体算法如下：

1. 输入：训练数据集

$$T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

其中 $x_i \in \mathcal{X} \subseteq \mathbb{R}^n$：n 维特征向量；$y_i \in \mathcal{Y} = \{-1, +1\}$：类别标签

2. 输出：最终分类器 $G(x)$

3. 算法流程：

1）初始化训练数据的权值分布，初始状态下每个样本的权重都是相等的 $1/N$

$$D_1 = (w_{11}, w_{12}, \ldots, w_{1N}), \quad w_{1i} = \frac{1}{N}, \ i = 1, 2, \ldots, N$$

2）对迭代的第 m 轮，$m = 1, 2, \ldots, M$，用有权值分布 D_m 的训练数据学习，得到基本分类器

$$G_m(x) : \mathcal{X} \to \{-1, +1\}$$

3）计算 $G_m(x)$ 在训练数据集上的分类误差率（样本权值分布有：$\sum_{i=1}^{N} W_{mi} = 1$）：

$$e_m = \sum_{i=1}^{N} P(G_m(x_i) \neq y_i) = \sum_{i=1}^{N} w_{mi} I(G_m(x_i) \neq y_i) = \sum_{G_m(x_i) \neq y_i} w_{mi}$$

这里 $I(\cdot)$ 是指示函数，即基本分类器与实际分类不同时（误分类）取 1，否则为 0.

4）计算 $G_m(x)$ 的系数，即该弱分类器的权重，错误率越低，权重越高，贡献越大：

$$\alpha_m = \frac{1}{2}\ln\frac{1-e_m}{e_m}$$

若 $e_m = 0.5$，$\alpha_m = 0$ 该分类器无贡献，等同随机猜测；若 $e_m < 0.5$，$\alpha_m > 0$ 错误率越低，α_m 越大，是有效分类器；$e_m > 0.5$，$\alpha_m < 0$ 是无效分类器。

5）更新训练数据集的权值分布 $D_{m+1} = (w_{m+1,1}, \ldots, w_{m+1,i}, \ldots, w_{m+1,N})$

$$w_{m+1,i} = \frac{w_{mi}}{Z_m}e^{-\alpha_m y_i G_m(x_i)}, \quad i = 1, 2, \ldots, N$$

$$Z_m = \sum_{i=1}^{N} w_{mi}e^{-\alpha_m y_i G_m(x_i)}$$

正确分类时，$y_i G_m(x_i) = 1$；错误分类时，$y_i G_m(x_i) = -1$；其中 Z_m 的作用是进行归一化，确保所有样本权值和为 1。上述可写为以下分段函数：

$$w_{m+1,i} = \begin{cases} \dfrac{w_{mi}}{Z_m}e^{-\alpha_m}, & G_m(x_i) = y_i \\ \dfrac{w_{mi}}{Z_m}e^{\alpha_m}, & G_m(x_i) \neq y_i \end{cases}$$

6）构建基本分类器的线性组合 $f(x) = \sum_{m=1}^{M}\alpha_m G_m(x)$ 得到最终分类器：

$$G(x) = \text{sign}(f(x)) = \text{sign}\left(\sum_{m=1}^{M}\alpha_m G_m(x)\right)$$

下图则展示了 AdaBoost 是如何将多个弱分类器组合成一个强分类器的过程。

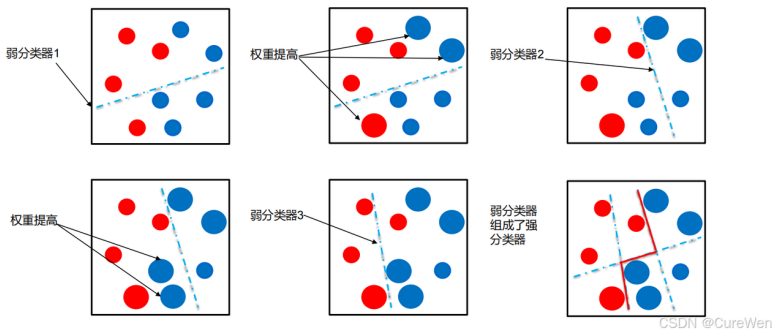


图 15: AdaBoost 图示

以如下数据进行 AdaBoost 具体运用过程示例展示：

序号	1	2	3	4	5	6	7	8	9	10
x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	−1	−1	−1	1	1	1	−1

初始化 $D_1=(w_{11},w_{12},\cdots w_{110})$ $w_{1i}=\frac{1}{10}$，$i=1,2,\cdots,10$

迭代：(i) $m=1$ 时，阈值取 2.5 分类误差率最小，有弱分类器

$$G_1(x)=\begin{cases}1, & x<2.5\\ -1, & x>2.5\end{cases}$$

$G_1(x)$ 误差率为：$e_1=\sum_{G_1(x_i)\neq y_i} w_{1i}=0.3$，系数 $\alpha_1=\frac{1}{2}\ln\frac{1-e_1}{e_1}=0.4236$

更新后权值分布：$D_2=(0.0715,0.0715,0.0715,0.0715,0.0715,0.0715,0.1666,0.1666,0.1666,0.0715)$

$\Rightarrow f_1(x)=0.4236G_1(x)$

(ii) $m=2$ 时，阈值取 8.5 时分类误差率最小

$$G_2=\begin{cases}1, & x<8.5\\ -1, & x>8.5\end{cases}$$

$\Rightarrow e_2=0.2143$；$\alpha_2=0.6496$

更新权值分布：$D_3=(0.0455,0.0455,0.0455,0.1667,0.1667,0.1667,0.1060,0.1060,0.1060,0.0455)$

$\Rightarrow f_2(x)=0.4236G_1(x)+0.6496G_2(x)$

(iii) $m=3$ 时，阈值取 5.5 分类误差率最低，最终得到 $\alpha_3=0.7514$（误分类点为0）

$\Rightarrow G(x)=\mathrm{sign}[f_3(x)]=\mathrm{sign}(0.4236G_1(x)+0.6496G_2(x)+0.7514G_3(x))$

8 统计决策理论和贝叶斯分类器

8.1 统计决策理论

先前的监督学习中，我们的目标是根据样本的输入特征可以对样本进行正确分类判断。但是现实的不确定性，我们无法无法百分百确定一个样本的真实类别，只能通过概率与统计规律，做出最优、最可靠的决策。**统计决策理论**就是在不确定环境下，利用概率知识、样本信息，指导我们如何选择最优分类规则的理论体系。

8.1.1 决策例子

假如我们要判断一个细胞是否为癌细胞，但是又**没有任何该细胞的检测信息**，此时就只能根据历史统计得到的**先验概率** $P(\omega_i)$，ω_1 代表正常细胞，ω_2 代表异常细胞。则可根据大量统计资料对某区域正常与异常细胞比例估计 $P(\omega_1)$ 与 $P(\omega_2)$

决策规则：

$$x\begin{cases}\in\omega_1 & 若P(\omega_1)\geq P(\omega_2)\\ \in\omega_2 & 若P(\omega_1)<P(\omega_2)\end{cases}$$

错误率：$P(\text{error})=1-P(\omega_1)=P(\omega_2)$ （如果决策$x\in\omega_1$）

而当我们知道细胞的检测信息 x 时，则是根据后验概率 $P(\omega_i|x)$（一种条件概率）判断，即在细胞特征 x 条件下，该样本属于类别 ω_i 的概率。

决策规则：

$$x\begin{cases}\in\omega_1 & 若P(\omega_1|x)\geq P(\omega_2|x)\\ \in\omega_2 & 若P(\omega_1|x)<P(\omega_2|x)\end{cases}$$

错误率：$P(\text{error}|x)=1-P(\omega_1|x)=P(\omega_2|x)$ （如果决策$x\in\omega_1$）

贝叶斯公式：$P(\omega_i|x)=\frac{P(\omega_i,x)}{P(x)}=\frac{P(x|\omega_i)P(\omega_i)}{P(x)}$

8.1.2 贝叶斯决策

写在前面，接下来的部分 P 代表概率，p 代表概率密度！

贝叶斯决策就是在**不完全情报下**，对部分未知的状态用主观概率估计，然后用**贝叶斯公式**对发生概率进行修正，最后再利用期望值和修正概率做出最优决策。**贝叶斯决策理论方法**是统计模型决策中的一个基本方法，有基本思想：

1. 已知**类条件概率密度参数表达式** $p(x|\omega_i)$（某类别下，特征 x 出现的概率）和先验概率 $P(\omega_i)$
2. 利用贝叶斯公式转换成**后验概率**
3. 根据后验概率大小进行决策分类

1）最小错误率贝叶斯决策：对于二分类问题，样本 x 上被分错的概率：

$$P(\text{error}|x)=\begin{cases}P(\omega_2|x) & x\in\omega_1\\ P(\omega_1|x) & x\in\omega_2\end{cases}$$

错误率：上述的 $P(\text{error}|x)$ 是单个样本的错误概率，我们关注的错误率是整个样本空间上的平均错误率：

$$P(\text{error})=\int P(\text{error}|x)p(x)\,dx$$

最小错误率贝叶斯决策即 $\min P(\text{error})=\min\int P(\text{error}|x)p(x)\,dx$，由于 $p(x)$ 是特征 x 固定的概率密度函数，因此这个最小化问题又等价于：

$$\min P(\text{error}|x)\quad\forall x$$

容易求解的**最优决策规则**是：

$$P(\omega_1|x)\gtrless P(\omega_2|x),\ \text{则}x\in\begin{cases}\omega_1\\ \omega_2\end{cases}$$

后验概率根据贝叶斯公式有：

$$P(\omega_i|x)=\frac{p(x|\omega_i)P(\omega_i)}{p(x)}=\frac{p(x|\omega_i)P(\omega_i)}{\sum_{j=1}^{2}p(x|\omega_j)P(\omega_j)},\quad i=1,2$$

Q：为啥这里的贝叶斯公式里面会有概率密度呢?

A：首先根据贝叶斯公式有 $P(\omega_i|x)=\frac{P(\omega_i,x)}{P(x)}=\frac{P(x|\omega_i)P(\omega_i)}{P(x)}$

$$\begin{cases}P(x|\omega_i)=\int_x^{x+\Delta x}p(\xi|\omega_i)\,d\xi\\ P(x)=\int_x^{x+\Delta x}p(\xi)\,d\xi\end{cases}\xrightarrow[\xi_1,\xi_2\in[x,x+\Delta x]]{\text{积分中值定理}}\begin{cases}p(\xi_1|\omega_i)\Delta x\\ p(\xi_2)\Delta x\end{cases}$$

当 $\Delta x\to 0$ 时，$\xi_1,\xi_2\to x$，因此

$$P(x|\omega_i)\to p(x|\omega_i)\Delta x$$

$$P(x)\to p(x)\Delta x$$

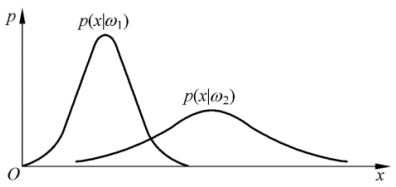
$$\Rightarrow\frac{P(x|\omega_i)P(\omega_i)}{P(x)}=\frac{p(x|\omega_i)\Delta x\,P(\omega_i)}{p(x)\Delta x}=\frac{p(x|\omega_i)P(\omega_i)}{p(x)}$$

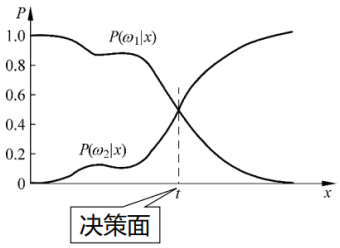
最优决策规则实际就是在比较 $P(\omega_1|x)$ 与 $P(\omega_2|x)$ 大小，因此作比有 $\frac{P(\omega_1|x)}{P(\omega_2|x)}=\frac{p(x|\omega_1)\cdot P(\omega_1)}{p(x|\omega_2)\cdot P(\omega_2)}\gtrless 1$，

写成似然比的形式，最优决策规则等价为：

$$若l(x)=\frac{p(x|\omega_1)}{p(x|\omega_2)}\gtrless\lambda=\frac{P(\omega_2)}{P(\omega_1)},\ 则x\in\begin{cases}\omega_1\\ \omega_2\end{cases}$$

如下两图则分别展示了类条件概率密度和后验概率图。





对于多类判别决策，则选取最大的后验概率作为决策判断依据。

$$P(\omega_i|x)=\max_{j=1,\cdots,c}P(\omega_j|x)\Rightarrow x\epsilon\omega_i$$

$$P(x|\omega_i)P(\omega_i)=\max_{j=1,\cdots,c}P(x|\omega_j)P(\omega_j)\Rightarrow x\epsilon\omega_i$$

如下是一个最小错误率贝叶斯决策例题：

> 假设在某个局部地区细胞识别中正常 (ω_1) 和异常 (ω_2) 两类的先验概率分别为
>
> 正常状态 $P(\omega_1)=0.9$
>
> 异常状态 $P(\omega_2)=0.1$
>
> 现有一待识别的细胞，其观察值为 x，从类条件概率密度曲线上分别查得
>
> $$p(x|\omega_1)=0.2,\quad p(x|\omega_2)=0.4$$
>
> 试对该细胞 x 进行分类。

利用贝叶斯公式，分别计算 ω_1 及 ω_2 的后验概率。

$$P(\omega_1|x)=\frac{P(x|\omega_1)\cdot P(\omega_1)}{P(x|\omega_1)\cdot P(\omega_1)+P(x|\omega_2)\cdot P(\omega_2)}=\frac{0.2\times0.9}{0.2\times0.9+0.1\times0.4}=0.818$$

$$P(\omega_2|x)=\frac{P(x|\omega_2)\cdot P(\omega_2)}{P(x|\omega_1)\cdot P(\omega_1)+P(x|\omega_2)\cdot P(\omega_2)}=\frac{0.1\times0.4}{0.2\times0.9+0.1\times0.4}=0.182.$$

$P(\omega_1|x)>P(\omega_2|x)$ 因此合理决策是将 x 归类于正常状态。

2）最小风险率贝叶斯决策：前面最小错误率贝叶斯决策我们默认所有错误的代价都是一样的，而去追求更高的正确率。但是实际生活中不同错误的后果往往是不同的，因此我们还需要将风险纳入考虑范围。

期望风险：采取某种决策 α_i（如判断为是/不是癌细胞）所带来的平均风险

$$R(\alpha_i|x) \equiv \sum_{j=1}^{c} \lambda(\alpha_i, \omega_j) P(\omega_j|x), \quad i = 1, \ldots, k$$

其中 α_i 是我们采取的第 i 个决策；ω_j 是第 j 个样本的真实状态；$\lambda(\alpha_i, \omega_j)$ 作为损失函数，在具体情况有固定的常数表。

由此最小风险率贝叶斯决策求解如下最优化问题：

$$\min R(\alpha) = \min \int R(\alpha(x)|x) p(x)\, dx$$

即 $\alpha = \arg\min_{i=1,\ldots,k} R(\alpha_i|x)$，对每个样本 x 计算所有决策的条件风险，选择条件风险最小的那个决策：

$$\text{若} \lambda_{11} P(\omega_1|x) + \lambda_{12} P(\omega_2|x) \lesseqgtr \lambda_{21} P(\omega_1|x) + \lambda_{22} P(\omega_2|x), \text{ 则} x \in \begin{cases} \omega_1 \\ \omega_2 \end{cases}$$

$$\text{若} l(x) = \frac{p(x|\omega_1)}{p(x|\omega_2)} \gtreqless \frac{P(\omega_2)}{P(\omega_1)} \cdot \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}}, \text{ 则} x \in \begin{cases} \omega_1 \\ \omega_2 \end{cases}$$

如下在最小错误率贝叶斯决策的例题下，考虑最小风险率贝叶斯问题：

决策	状态	
	ω_1	ω_2
α_1	0	6
α_2	1	0

表 1: 损失矩阵

已知 $P(\omega_1)=0.9$，$P(\omega_2)=0.1$，$p(x|\omega_1)=0.2$，$p(x|\omega_2)=0.4$

且有后验概率为 $P(\omega_1|x)=0.818$，$P(\omega_2|x)=0.182$

计算条件风险：$R(\alpha_1|x)=\sum_{j=1}^{2}\lambda_{1j}P(\omega_j|x)=\lambda_{12}P(\omega_2|x)=1.092$

$R(\alpha_2|x)=\lambda_{21}P(\omega_1|x)=0.818$

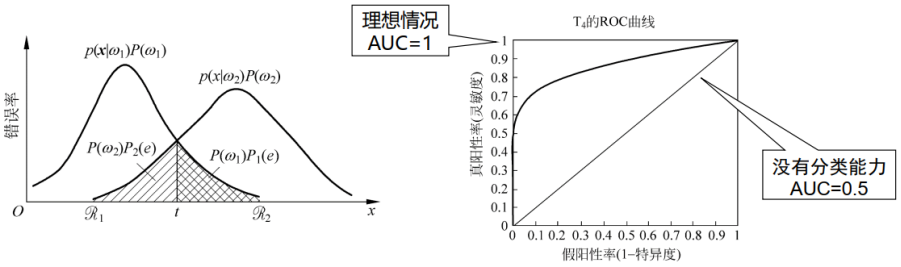
由于 $R(\alpha_1|x) > R(\alpha_2|x)$

决策 ω_2 的条件风险小于决策 ω_1 的条件风险，因此选取 α_2，将细胞 x 识别为 ω_2 ——异常细胞.

8.1.3 二分类问题评价指标

状态 / 决策	阳性	阴性
阳性	真阳性 (TP)	假阳性 (FP)
阴性	假阴性 (FN)	真阴性 (TN)

- 第一类错误率：$\frac{FP}{FP+TN}$；第二类错误率：$\frac{FN}{FN+TP}$
- 灵敏度（召回率 Rec）：$S_n = \frac{TP}{TP+FN}$；特异度：$S_p = \frac{TN}{TN+FP}$
- 正确率：$\mathrm{Acc} = \frac{TP+TN}{TP+TN+FP+FN}$；精确率：$\mathrm{Pre} = \frac{TP}{TP+FP}$；F 度量：$F = \frac{2\,\mathrm{Rec}\cdot\mathrm{Pre}}{\mathrm{Rec}+\mathrm{Pre}}$
- 采用不同的阈值，可以使第一类错误率和第二类错误率连续变化。**ROC 曲线**（受试者工作特征曲线），展示了**阈值移动**时两类错误率的变化，越靠近左上角的点，性能越好，曲线越靠近左上角，模型的整体分类能力越强。**AUC（曲线下面积）**是 ROC 曲线和横轴围成的面积，它是一个 0 到 1 之间的数，量化模型的整体分类能力。



8.2 概率密度函数的估计

概率密度函数估计的方法：

- 参数估计：已知概率密度函数形式，估计未知或部分未知参数，如点估计、极大似然估计。
- 非参数估计：概率密度函数形式未知，直接估计概率密度函数，如核密度估计、直方图估计。

8.2.1 最大似然估计

在概统中已经接触过最大似然估计（MLE），此处略微回顾一下。

基本假设：待估参数 θ 未知但确定；样本独立同分布；概率密度函数形式已知；不同类别的参数独立，可分别处理。

样本：$X = \{x_1, x_2, \ldots, x_N\}$ **似然函数**：$L(\theta) = p(\mathcal{X}|\theta) = p(x_1, x_2, \ldots, x_3|\theta) = \prod_{i=1}^{N} p(x_i|\theta)$

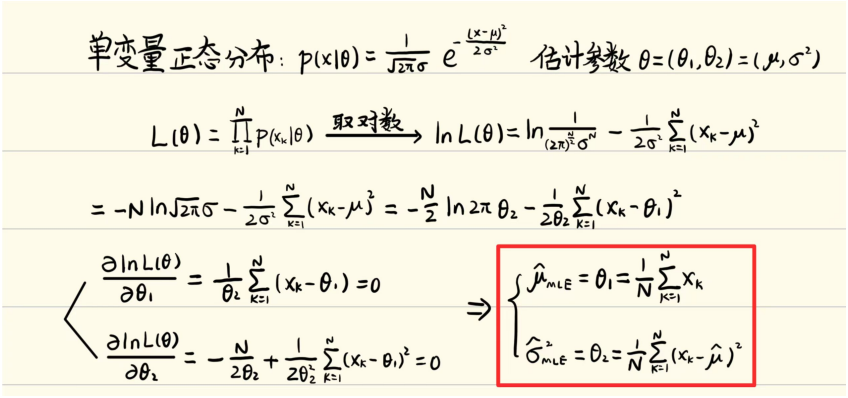
似然函数连续可微，转换为求解：$\frac{dL(\theta)}{d\theta} = 0$

一般地，未知参数为多个：$\hat{\theta} = \left[\theta_1, \theta_2, \ldots, \theta_s\right]^T$ 则是分别对每个参数求偏导等于 0.

$$\nabla_\theta = \left[\frac{\partial}{\partial\theta_1}, \frac{\partial}{\partial\theta_2}, \cdots, \frac{\partial}{\partial\theta_s}\right]^T$$

$$\nabla_\theta L(\theta) = 0$$

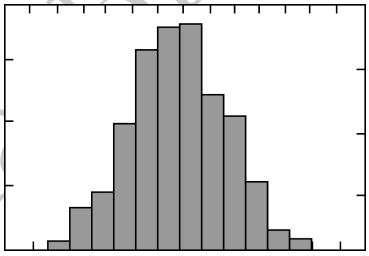
如下是对正态分布下的最大似然估计求解：



8.2.2 直方图法

当需要估计的概率密度函数的形式未知时，无法使用参数估计。选取非参数估计方法，需要注意非参数方法的共同问题是对样本数量需求较大，只要样本数目足够大众可以保证收敛于任何复杂的位置密度，但是计算量和存储量都比较大。

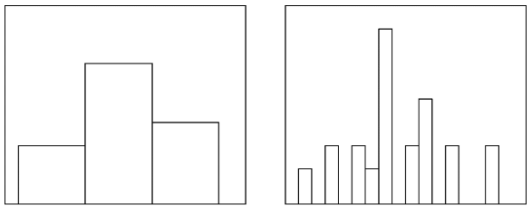
直方图法把样本 x 每个分量分成 k 个等间隔小舱，每个小舱的体积为 V，统计落入小舱内的样本数目 q_i，把每个小舱的概率密度看作常数，估计值为 $\frac{q_i}{NV}$（N 是样本总数）



- **基本原理**：区间体积 V 足够小时，可以假定该小范围内 $p(x)$ 是常数，随机向量落入小区域范围的估计为：$P_R=\int_R p(x)\,dx=p(x)V$，小区域中落入 k 个样本时 P_R 的估计为：$\hat{P}_R=\frac{k}{N}$，直方图小舱内的概率密度估计为：
$$\hat{p}(x)=\frac{k}{NV}$$
- **小舱选择**：小舱过大会导致估计的密度函数粗糙；小舱过小会使估计结果不连续。样本趋于无穷多时的收敛条件为：$\lim_{n\to\infty}V_n=0$，$\lim_{n\to\infty}k_n\to\infty$，$\lim_{n\to\infty}\frac{k_n}{n}=0$

8.3 朴素贝叶斯

在先前的贝叶斯决策中，涉及到类条件概率 $P(x|\omega_i)$，即在类别 ω_i 下，特征 x 出现的概率。当 x 包含特征个数很多如 10 个时，每个特征都有“是”/“否”两种取值，这样导致特征的组合数高达 2^{10} 种。条件概率为指数级别的参数 $K\prod_{j=1}^{n}S_j$ 无法用有限样本进行参数估计，**朴素贝叶斯**则解决了这个问题。



(a) 小窗过宽 (b) 小窗过窄

- 特征为 X、类别为 Y，训练数据集：$T=\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$；学习先验概率分布：$P(Y=c_k),\ k=1,2,\ldots,K$
- **条件独立性假设**：固定类别 $Y=c_k$ 下，所有特征 $X^{(1)},X^{(2)},\ldots,X^{(n)}$ 之间相互独立，得到

$$\begin{aligned}P(X=x\mid Y=c_k)&=P\left(X^{(1)}=x^{(1)},\ldots,X^{(n)}=x^{(n)}\mid Y=c_k\right)\\&=\prod_{j=1}^{n}P\left(X^{(j)}=x^{(j)}\mid Y=c_k\right)\end{aligned}$$

这个很强的假设，牺牲了分类准确性，“朴素”贝叶斯名字由来。

得到贝叶斯定理：

$$P(Y=c_k\mid X=x)=\frac{P(X=x\mid Y=c_k)P(Y=c_k)}{\sum_k P(X=x\mid Y=c_k)P(Y=c_k)}=\frac{P(Y=c_k)\prod_j P\left(X^{(j)}=x^{(j)}\mid Y=c_k\right)}{\sum_k P(Y=c_k)\prod_j P\left(X^{(j)}=x^{(j)}\mid Y=c_k\right)}$$

最小错误率贝叶斯分类器**最大化后验概率**，有最大化后验概率的 c_k 类别即为最后的决策类别：

$$y=f(x)=\arg\max_{c_k}\frac{P(Y=c_k)\prod_j P\left(X^{(j)}=x^{(j)}\mid Y=c_k\right)}{\sum_k P(Y=c_k)\prod_j P\left(X^{(j)}=x^{(j)}\mid Y=c_k\right)}$$

分母对所有 c_k 相同，等价于：

$$y=f(x)=\arg\max_{c_k}P(Y=c_k)\prod_j P\left(X^{(j)}=x^{(j)}\mid Y=c_k\right)$$

于是最后的问题就是估计 $P(Y=c_k)$ 和 $P\left(X^{(j)}=x^{(j)}\mid Y=c_k\right)$

先验概率 $P(Y=c_k)$ 的极大似然估计：

$$P(Y=c_k)=\frac{\sum_{i=1}^{N}I(y_i=c_k)}{N},\quad k=1,2,\ldots,K$$

条件概率 $P\left(X^{(j)}=x^{(j)}\mid Y=c_k\right)$ 的极大似然估计：

$$P\left(X^{(j)}=a_{jl}\mid Y=c_k\right)=\frac{\sum_{i=1}^{N}I\left(X_i^{(j)}=a_{jl},y_i=c_k\right)}{\sum_{i=1}^{N}I\left(y_i=c_k\right)}$$

注意：运用上述极大似然估计时可能出现所要估计的概率值为 0 的情况，即如果某个特征取值 a_{jl} 在训练集中未出现在类别 c_k 下，那么此时上面公式得到的 $P\left(X^{(j)}=a_{jl}\mid Y=c_k\right)$ 值为 0，而朴素贝叶

斯里最大化后验概率中是连乘形式，会导致整个后验概率全部为 0。因此应用贝叶斯估计法（拉普拉斯平滑）估计相应的概率，给计数增加常数项，保证概率不为 0.

先验概率：

$$P(Y=c_k)=\frac{\sum_{i=1}^{N}I(y_i=c_k)+\lambda}{N+K\lambda}$$

条件概率：

$$P\left(X^{(j)}=a_{jl}\mid Y=c_k\right)=\frac{\sum_{i=1}^{N}I\left(X_i^{(j)}=a_{jl},y_i=c_k\right)+\lambda}{\sum_{i=1}^{N}I\left(y_i=c_k\right)+S_j\lambda}$$

这里 K 是类别总数，即一共多少种 c_k 可能；N 是训练集的总样本数；
S_j 是第 j 个特征的可能取值总数；a_{jl} 是第 j 个特征的第 l 个取值

朴素贝叶斯示例题：

例. 试由下面训练数据学习一个朴素贝叶斯分类器并确定 $x=(2,S)^{\mathrm{T}}$ 的类标记 y。表中 $X^{(1)},X^{(2)}$ 为特征，取值的集合分别为 $A_1=\{1,2,3\}$，$A_2=\{S,M,L\}$，Y 为类标记，$Y\in C=\{1,-1\}$。

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$X^{(1)}$	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
$X^{(2)}$	S	M	M	S	S	S	M	M	L	L	L	M	M	L	L
Y	-1	-1	1	1	-1	-1	-1	1	1	1	1	1	1	1	-1

解：根据算法，计算下列概率：

$$P(Y=1)=\frac{9}{15},\quad P(Y=-1)=\frac{6}{15}$$

$$P(X^{(1)}=1\mid Y=1)=\frac{2}{9},\quad P(X^{(1)}=2\mid Y=1)=\frac{3}{9},\quad P(X^{(1)}=3\mid Y=1)=\frac{4}{9}$$
$$P(X^{(2)}=S\mid Y=1)=\frac{1}{9},\quad P(X^{(2)}=M\mid Y=1)=\frac{4}{9},\quad P(X^{(2)}=L\mid Y=1)=\frac{4}{9}$$

$$P(X^{(1)}=1\mid Y=-1)=\frac{3}{6},\quad P(X^{(1)}=2\mid Y=-1)=\frac{2}{6},\quad P(X^{(1)}=3\mid Y=-1)=\frac{1}{6}$$
$$P(X^{(2)}=S\mid Y=-1)=\frac{3}{6},\quad P(X^{(2)}=M\mid Y=-1)=\frac{2}{6},\quad P(X^{(2)}=L\mid Y=-1)=\frac{1}{6}$$

对于给定的 $x=(2,S)^{\mathrm{T}}$，计算：

$$P(Y=1)P(X^{(1)}=2\mid Y=1)P(X^{(2)}=S\mid Y=1)=\frac{9}{15}\cdot\frac{3}{9}\cdot\frac{1}{9}=\frac{1}{45}$$
$$P(Y=-1)P(X^{(1)}=2\mid Y=-1)P(X^{(2)}=S\mid Y=-1)=\frac{6}{15}\cdot\frac{2}{6}\cdot\frac{3}{6}=\frac{1}{15}$$

由于 $P(Y=-1)P(X^{(1)}=2\mid Y=-1)P(X^{(2)}=S\mid Y=-1)$ 最大，所以 $y=-1$。

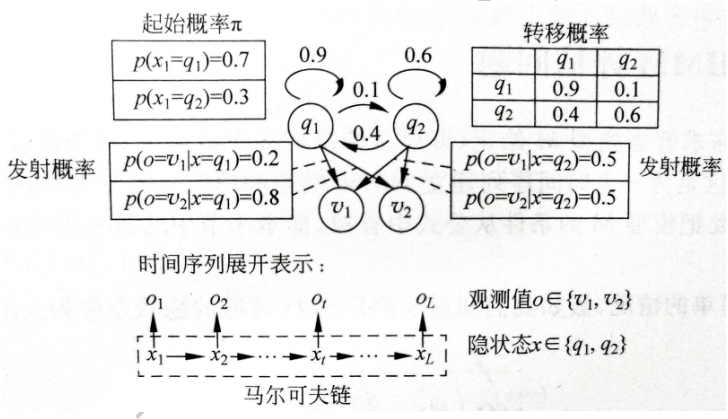
9 隐马尔可夫模型（HMM）

马尔可夫链是一个系统的状态随时间的变化，且下一个状态只和当前状态有关，和更早的状态无关。马尔可夫模型所有的状态都是可见的，我们能直接观测到每个时刻的状态，只需要建模状态之间的跳转关系。如下马尔可夫状态表，展示城市的天气变化，即当今天天气为 XX 时，明天天气是 XX 的概率。

今天天气\明天天气	晴天(sunny)	有雨(rainy)	多云(cloudy)	有雾(foggy)
晴天(sunny)	0.6	0.1	0.2	0.1
有雨(rainy)	0.5	0.2	0.2	0.1
多云(cloudy)	0.1	0.7	0.1	0.1
有雾(foggy)	0.0	0.3	0.4	0.3

9.1 隐马尔可夫模型的定义

隐马尔可夫模型是关于时序/序列的概率模型，由一个隐藏的马尔科夫链随机生成不可观测的状态随机序列（**隐状态序列**），再由各个状态生成一个观测从而产生观测随机序列（**观测序列**）的过程。序列的每一个位置又可以看作是一个时刻。



组成：

- 所有可能状态的集合：$Q=\{q_1, q_2, \ldots, q_N\}$
- 所有可能观测的集合：$V=\{v_1, v_2, \ldots, v_M\}$
- 长度为 T 的状态序列：$I=(i_1, i_2, \ldots, i_T)$
- 对应的观测序列：$O=(o_1, o_2, \ldots, o_T)$

三要素——$\lambda=(A, B, \pi)$

初始概率分布：$\pi=(\pi_1, \pi_2, \ldots, \pi_N)$；有 $\pi_k=P(i_1=q_k), \quad k=1, 2, \ldots, N$

状态转移概率矩阵 A：

$$A=[a_{ij}]_{N\times N}$$

$$a_{ij}=P(i_{t+1}=q_j \mid i_t=q_i), \quad i=1, 2, \ldots, N;\ j=1, 2, \ldots, N$$

即时刻 t 处于状态 q_i 的条件下在时刻 $t+1$ 转移到状态 q_j 的概率。

观测概率矩阵 B：

$$B = [b_j(k)]_{N\times M}$$

$$b_j(k) = P(o_t = v_k \mid i_t = q_j), \quad k = 1,2,\dots,M,\ j = 1,2,\dots,N$$

即时刻 t 处于 q_j 的条件下生成观测 v_k 的概率。

两个基本假设：

1. **齐次马尔科夫性假设**：隐马尔可夫链中，时刻 t 的状态只和时刻 $t-1$ 状态有关，即

$$P(i_t \mid i_{t-1}, o_{t-1}, \dots, i_1, o_1) = P(i_t \mid i_{t-1}), \quad t = 1,2,\dots,T$$

2. **观测独立性假设**：时刻 t 的观测只和当前时刻 t 的状态有关，即

$$P(o_t \mid i_T, o_T, i_{T-1}, o_{T-1}, \dots, i_{t+1}, o_{t+1}, i_t, i_{t-1}, o_{t-1}, \dots, i_1, o_1) = P(o_t \mid i_t)$$

例. 有 4 个盒子，每个盒子装有红、白两种球，红、白球数分别为：盒 1（红 5、白 5）、盒 2（红 3、白 7）、盒 3（红 6、白 4）、盒 4（红 8、白 2）。按以下规则抽球：

1. 初始从 4 个盒子中等概率选取 1 个，抽 1 球记录颜色后放回；
2. 按规则转移到下一盒：盒 1 必转盒 2，盒 2/3 分别以 0.4/0.6 概率转左/右盒，盒 4 以 0.5 概率留盒 4 或转盒 3；
3. 在新盒中抽球记录颜色后放回，重复上述转移与抽球操作共 5 次，得到观测序列 O = (红, 红, 白, 白, 红)；

观察者仅能观测球色序列，无法观测盒子序列。

解：盒子对应状态，状态的集合是

$$Q = \{\text{盒子1},\ \text{盒子2},\ \text{盒子3},\ \text{盒子4}\}, \quad N = 4$$

球的颜色对应观测，观测的集合是

$$V = \{\text{红},\ \text{白}\}, \quad M = 2$$

状态序列和观测序列长度 $T = 5$。

初始概率分布为

$$\pi = (0.25,\ 0.25,\ 0.25,\ 0.25)^{\mathrm{T}}$$

状态转移概率分布为

$$A=\begin{bmatrix}0 & 1 & 0 & 0\\ 0.4 & 0 & 0.6 & 0\\ 0 & 0.4 & 0 & 0.6\\ 0 & 0 & 0.5 & 0.5\end{bmatrix}$$

观测概率分布为

$$B=\begin{bmatrix}0.5 & 0.5\\ 0.3 & 0.7\\ 0.6 & 0.4\\ 0.8 & 0.2\end{bmatrix}$$

观测序列的生成:

输入：隐马尔可夫模型 $\lambda=(A,B,\pi)$，观测序列长度 T。

输出：观测序列 $O=(o_1,o_2,\dots,o_T)$。

1. 按照初始状态分布 π 产生状态 i_1;
2. 令 $t=1$;
3. 按照状态 i_t 的观测概率分布 $b_{i_t}(k)$ 生成 o_t;
4. 按照状态 i_t 的状态转移概率分布 $\{a_{i_t i_{t+1}}\}$ 产生状态 i_{t+1}，$i_{t+1}=1,2,\dots,N$;
5. 令 $t=t+1$，如果 $t<T$，转步骤 (3)；否则，终止。

三个基本问题:

1. 概率计算问题——给定模型参数,计算某一观测序列出现的概率:给定 $\lambda=(A,B,\pi)\quad O=(o_1,o_2,\dots,o_T)$，计算 $P(O\mid\lambda)$;
2. 预测问题(解码)——给定模型参数,给定某一观测序列,解码隐藏的状态序列,已知 $\lambda=(A,B,\pi)\quad O=(o_1,o_2,\dots,o_T)$，计算使 $P(I\mid O)$ 最大的状态序列 $I=(i_1,i_2,\dots,i_T)$;
3. 学习问题(最大似然参数拟合)——给定一系列观测序列,求解模型参数,已知 $O=(o_1,o_2,\dots,o_T)$，估计 $\lambda=(A,B,\pi)$，使 $P(O\mid\lambda)$ 最大。

9.2 概率计算问题

要计算给定模型和观测序列下的观测序列出现的概率，有多种方法可以计算。

9.2.1 直接计算

- 列举所有可能的长度为 T 状态序列 $I=(i_1,i_2,\dots,i_T)$
- 求各个状态序列 I 与观测序列 $O=(o_1,o_2,\dots,o_T)$ 的联合概率 $P(O,I\mid\lambda)$
- 然后对所有可能的状态序列求和，得到 $P(O\mid\lambda)$

模型下状态序列 $I=(i_1,i_2,\dots,i_T)$ 的概率为：$P(I\mid\lambda)=\pi_{i_1}a_{i_1i_2}a_{i_2i_3}\cdots a_{i_{T-1}i_T}$
在固定状态序列 I 观测到序列 $O=(o_1,o_2,\dots,o_T)$ 的概率：$P(O\mid I,\lambda)=b_{i_1}(o_1)b_{i_2}(o_2)\cdots b_{i_T}(o_T)$
O 和 I 同时出现的联合概率为：

$$P(O,I\mid\lambda)=P(O\mid I,\lambda)P(I\mid\lambda)=\pi_{i_1}b_{i_1}(o_1)a_{i_1i_2}b_{i_2}(o_2)\cdots a_{i_{T-1}i_T}b_{i_T}(o_T)$$

所有可能的状态序列 I 求和得到观测 O 的概率：

$$P(O\mid\lambda)=\sum_I P(O\mid I,\lambda)P(I\mid\lambda)=\sum_{i_1,i_2,\dots,i_T}\pi_{i_1}b_{i_1}(o_1)a_{i_1i_2}b_{i_2}(o_2)\cdots a_{i_{T-1}i_T}b_{i_T}(o_T)$$

直接计算是对所有可能的状态序列求和，共有 T 个时刻，每个时刻都有 N 个状态选择，时间复杂度为 $O(TN^T)$，**太过复杂**。

9.2.2 前向算法

在给定隐马尔科夫模型 λ 下，定义到时刻 t 部分观测序列为 $o_1,o_2,\dots,o_t$，且状态为 q_i 的概率为**前向概率**，记作

$$\alpha_t(i)=P(o_1,o_2,\dots,o_t,i_t=q_i\mid\lambda)$$

1）初始化前向概率：

$$\begin{aligned}\alpha_1(i)&=P(o_1,i_1=q_i\mid\lambda)\\&=P(o_1\mid i_1=q_i,\lambda)P(i_1=q_i)\\&=\pi_ib_i(o_1),\quad i=1,2,\dots,N\end{aligned}$$

2）递推公式：对于 $\forall t=1,2,\dots,T-1$ 有：

$$\begin{aligned}\alpha_{t+1}(i)&=P(o_1,o_2,\dots,o_{t+1},i_{t+1}=q_i\mid\lambda)\\&\overset{\text{全概率公式}}{=}\sum_{j=1}^N P(o_1,o_2,\dots,o_{t+1},i_t=q_j,i_{t+1}=q_i\mid\lambda)\\&\overset{\text{条件概率公式}}{=}\sum_{j=1}^N P(o_1,\dots,o_t,i_t=q_j\mid\lambda)\cdot P(i_{t+1}=q_i\mid o_1,\dots,o_t,i_t=q_j,\lambda)\cdot\\&\quad P(o_{t+1}\mid o_1,\dots,o_t,i_t=q_j,i_{t+1}=q_i,\lambda)\\&\overset{\text{两个基本假设}}{=}\sum_{j=1}^N P(o_1,\dots,o_t,i_t=q_j\mid\lambda)\cdot P(i_{t+1}=q_i\mid i_t=q_j,\lambda)\cdot P(o_{t+1}\mid i_{t+1}=q_i,\lambda)\\&=\sum_{j=1}^N\alpha_t(j)a_{ji}b_i(o_{t+1}),\quad i=1,2,\dots,N\end{aligned}$$

3）终止式：

$$\begin{aligned} P(O \mid \lambda) &= P(o_1, o_2, \ldots, o_T \mid \lambda) \\ &= \sum_{i=1}^{N} P(o_1, o_2, \ldots, o_T, i_T = q_i \mid \lambda) \\ &= \sum_{i=1}^{N} \alpha_T(i) \end{aligned}$$

利用递推公式，每次计算都可以使用上一次的结果，计算复杂度变成 $O(TN^2)$，大大简化了运算量。

例. 考虑盒子和球模型 $\lambda = (A, B, \pi)$，状态集合 $Q = \{1, 2, 3\}$，观测集合 $V = \{\text{红}, \text{白}\}$，

$$A = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5 & 0.5 \\ 0.4 & 0.6 \\ 0.7 & 0.3 \end{bmatrix}, \quad \pi = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \end{bmatrix}$$

设 $T = 3$，$O = (\text{红}, \text{白}, \text{红})$，试用前向算法计算 $P(O|\lambda)$。

1). 初值计算：$\alpha_1(1) = \pi_1 b_1(o_1) = 0.10$ ； $\alpha_1(2) = \pi_2 b_2(o_1) = 0.16$ ； $\alpha_1(3) = \pi_3 b_3(o_1) = 0.28$

2). 递推计算：$\alpha_2(1) = \sum_{j=1}^{3} \alpha_1(j) a_{j1} b_1(o_2) = 0.154 \times 0.5 = 0.077$　　$\alpha_3(1) = \sum_{j=1}^{3} \alpha_2(j) a_{j1} b_1(o_3) = 0.04187$

$\alpha_2(2) = \sum_{j=1}^{3} \alpha_1(j) a_{j2} b_2(o_2) = 0.184 \times 0.6 = 0.1104$　　$\alpha_3(2) = \sum_{j=1}^{3} \alpha_2(j) a_{j2} b_2(o_3) = 0.03551$

$\alpha_2(3) = \sum_{j=1}^{3} \alpha_1(j) a_{j3} b_3(o_2) = 0.202 \times 0.3 = 0.0606$　　$\alpha_3(3) = \sum_{j=1}^{3} \alpha_2(j) a_{j3} b_3(o_3) = 0.05284$

3). 终止：$P(O|\lambda) = \sum_{i=1}^{3} \alpha_3(i) = 0.13022$

9.2.3 后向算法

后向算法是前向算法的对称形式，前向算法是从前向后遍历，后向算法是从后向前遍历。给定隐马尔科夫模型 λ 下，定义在时刻 t 状态为 q_i 的条件下，从 $t+1$ 到 T 的部分观测序列为 o_{t+1}，o_{t+2}，$\ldots$，o_T 的概率为后向概率，记作

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T \mid i_t = q_i, \lambda), \quad \forall t \le T-1, \quad \beta_T(i) = 1$$

1）初始化：最后一个时刻 T 不存在后续观测，因此无论处于哪个状态 i，生成的空序列的概率都为 1。

$$\beta_T(i) = 1, \quad i = 1, 2, \ldots, N$$

2）递推公式：$\forall t = T-1, T-2, \ldots, 1$ 有：

$$
\begin{aligned}
\beta_t(i) &= P(o_{t+1}, \ldots, o_T \mid i_t = q_i, \lambda) \\
&\overset{\text{全概率公式}}{=} \sum_{j=1}^{N} P(o_{t+1}, \ldots, o_T, i_{t+1} = q_j \mid i_t = q_i, \lambda) \\
&\overset{\text{条件概率公式}}{=} \sum_{j=1}^{N} P(o_{t+1}, \ldots, o_T \mid i_{t+1} = q_j, i_t = q_i, \lambda) P(i_{t+1} = q_j \mid i_t = q_i, \lambda) \\
&\overset{\text{两个基本假设}}{=} \sum_{j=1}^{N} P(o_{t+1}, \ldots, o_T \mid i_{t+1} = q_j, \lambda) P(i_{t+1} = q_j \mid i_t = q_i, \lambda) \\
&= \sum_{j=1}^{N} P(o_{t+2}, \ldots, o_T \mid i_{t+1} = q_j, \lambda) P(o_{t+1} \mid i_{t+1} = q_j, \lambda) P(i_{t+1} = q_j \mid i_t = q_i, \lambda) \\
&= \sum_{j=1}^{N} \beta_{t+1}(j) b_j(o_{t+1}) a_{ij}, \quad i = 1, 2, \ldots, N
\end{aligned}
$$

3）终止：

$$
\begin{aligned}
P(O \mid \lambda) &= P(o_1, o_2, \ldots, o_T \mid \lambda) \\
&= \sum_{i=1}^{N} P(o_1, i_1 = q_i, o_2, \ldots, o_T \mid \lambda) \\
&= \sum_{i=1}^{N} P(i_1 = q_i \mid \lambda) P(o_1 \mid i_1 = q_i, \lambda) P(o_2, o_3, \ldots, o_T \mid i_1 = q_i, \lambda) \\
&= \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i)
\end{aligned}
$$

计算复杂度同样变成 $O(TN^2)$。

9.2.4 前向后向统一

利用前向概率和后向概率

原式：

$$
\begin{aligned}
P(O|\lambda) = P(o_1, \ldots, o_T|\lambda) &= \sum_{i=1}^{N} \sum_{j=1}^{N} P(o_1, \ldots, o_t, o_{t+1}, \ldots, o_T, i_t = q_i, i_{t+1} = q_j|\lambda) \\
&\overset{\text{条件概率}}{=} \sum_{i=1}^{N} \sum_{j=1}^{N} P(o_1, \ldots, o_t, i_t = q_i \mid i_{t+1} = q_j, \lambda)\, P(o_{t+1}, \ldots, o_T, i_{t+1} = q_j|\lambda)
\end{aligned}
$$

后部分：

$$\begin{aligned}P(o_{t+1},\dots,o_T,i_{t+1}=q_j|\lambda) &= P(o_{t+2},\dots,o_T \mid i_{t+1}=q_j,\lambda)\,P(o_{t+1},i_{t+1}=q_j|\lambda)\\ &= \beta_{t+1}(j)\,P(o_{t+1}\mid i_{t+1}=q_j,\lambda)\,P(i_{t+1}=q_j|\lambda)\\ &= \beta_{t+1}(j)\,b_j(o_{t+1})\,P(i_{t+1}=q_j|\lambda)\end{aligned}$$

前部分：

$$\begin{aligned}P(o_1,\dots,o_t,i_t=q_i\mid i_{t+1}=q_j,\lambda) &= \frac{P(o_1,\dots,o_t,i_t=q_i,i_{t+1}=q_j|\lambda)}{P(i_{t+1}=q_j|\lambda)}\\ &= \frac{P(i_{t+1}=q_j\mid o_1,\dots,o_t,i_t=q_i,\lambda)\,P(o_1,\dots,o_t,i_t=q_i|\lambda)}{P(i_{t+1}=q_j|\lambda)}\\ &= \frac{P(i_{t+1}=q_j\mid i_t=q_i,\lambda)\,P(o_1,\dots,o_t,i_t=q_i|\lambda)}{P(i_{t+1}=q_j|\lambda)}\\ &= \frac{a_{ij}\cdot\alpha_t(i)}{P(i_{t+1}=q_j|\lambda)}\end{aligned}$$

$$\therefore\ P(O|\lambda)=\sum_{i=1}^{N}\sum_{j=1}^{N}a_{ij}\,\alpha_t(i)\,\beta_{t+1}(j)\,b_j(o_{t+1})\quad t=1,2,\dots,T-1$$

特殊地有：

$$\begin{aligned}P(O|\lambda) &= P(o_1,\dots,o_T|\lambda)\\ &= \sum_{i=1}^{N}P(o_1,\dots,o_t,\dots,o_T,\ i_t=q_i|\lambda)\\ &= \sum_{i=1}^{N}P(o_{t+1},\dots,o_T\mid o_1,\dots,o_t,\ i_t=q_i,\lambda)\,P(o_1,\dots,o_t,\ i_t=q_i|\lambda)\\ &= \sum_{i=1}^{N}P(o_{t+1},\dots,o_T\mid i_t=q_i,\lambda)\,P(o_1,\dots,o_t,\ i_t=q_i|\lambda)\\ &= \sum_{i=1}^{N}\beta_t(i)\,\alpha_t(i),\quad t=1,2,\dots,T\end{aligned}$$

9.3 预测问题

预测问题是已知模型 $\lambda=(A,B,\pi)$ 和观测序列 $O=(o_1,o_2,\dots,o_T)$ 时，计算使后验概率 $P(I|O)$ 最大的状态序列 $I=(i_1,i_2,\dots,i_T)$

9.3.1 近似算法

近似算法的想法是，在每个时刻 t 选择在该时刻最有可能出现的状态 i_t^*，从而得到一个状态序列 $I=(i_1^*,i_2^*,\dots,i_T^*)$，将它作为预测结果，在 t 时刻处于状态 q_i 的概率为：

$$
\begin{aligned}
\gamma_t(i) &= P(i_t = q_i \mid O, \lambda) \\
&= \frac{P(o_1, \ldots, o_T, i_t = q_i \mid \lambda)}{P(O \mid \lambda)} \\
&= \frac{P(o_1, \ldots, o_t, i_t = q_i \mid o_{t+1}, \ldots, o_T, \lambda) P(o_{t+1}, \ldots, o_T \mid o_1, \ldots, o_t, i_t = q_i, \lambda)}{P(O \mid \lambda)} \\
&= \frac{P(o_1, \ldots, o_t, i_t = q_i \mid \lambda) \sum_{j=1}^{N} P(o_{t+1}, \ldots, o_T, i_{t+1} = q_j \mid o_1, \ldots, o_t, i_t = q_i, \lambda)}{P(O \mid \lambda)} \\
&= \frac{P(o_1, \ldots, o_t, i_t = q_i \mid \lambda) \sum_{j=1}^{N} P(o_{t+1}, \ldots, o_T, i_{t+1} = q_j \mid i_t = q_i, \lambda)}{P(O \mid \lambda)} \\
&= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}
\end{aligned}
$$

- 在每一时刻 t 最有可能的状态是：$i_t^* = \arg\max_{1 \le i \le N} [\gamma_t(i)], \quad t = 1, 2, \ldots, T$
- 从而得到状态序列：$I = (i_1^*, i_2^*, \ldots, i_T^*)$
- 得到的状态有可能实际不发生

9.3.2 维特比算法

维特比算法是采用动态规划解决预测问题，即求最优路径（一条路径对应着一个状态序列）。**原理**：如果最优路径在时刻 t 通过结点 i_t^*，那么这一路径从结点 i_t^* 到终点 i_T^* 的部分路径，对于从 i_t^* 到 i_T^* 的所有可能的部分路径来说，必须是最优的。（即最优路径是全局最优解，其子路径也是节点间的最优路径）

- 从时刻 $t = 1$ 开始，递推地计算在时刻 t 状态为 i 的各条部分路径的最大概率；
- 直至得到时刻 $t = T$ 时状态为 i 的各条路径的最大概率；
- 时刻 $t = T$ 的最大概率即为最优路径的概率 P^*，最优路径的终结点 i_T^* 也同时得到；
- 之后，为了找出最优路径的各个结点，从终结点 i_T^* 开始，由后向前逐步求得结点 $i_{T-1}^*, i_{T-2}^*, \ldots, i_1^*$，得到最优路径 $I^* = (i_1^*, i_2^*, \ldots, i_T^*)$

在时刻 t 状态为 i 的所有单个路径 $(i_1, i_2, \ldots, i_{t-1}, i)$ 中概率最大值为：

$$
\delta_t(i) = \max_{i_1, i_2, \ldots, i_{t-1}} P(i_t = i, i_{t-1}, \ldots, i_1, o_t, \ldots, o_1 \mid \lambda), \quad i = 1, 2, \ldots, N
$$

在时刻 t 状态为 i 的所有路径 $(i_1, i_2, \ldots, i_{t-1}, i)$ 中概率最大的路径的第 $t-1$ 个结点为：

$$
\Psi_t(i) = \arg \max_{1 \le j \le N} [\delta_{t-1}(j) \cdot a_{ji}], \quad i = 1, 2, \ldots, N
$$

1）初始化：

$$\delta_1(i) = P(i_1 = i, o_1 \mid \lambda) = P(o_1 \mid i_1 = i, \lambda)P(i_1 = i \mid \lambda) = \pi_i b_i(o_1)$$
$$\Psi_1(i) = 0, \quad i = 1, 2, \ldots, N$$

2）递推：对 $t = 2, 3, \ldots, T$

$$\begin{aligned}
\delta_{t+1}(i) &= \max_{i_1,\ldots,i_t} P(i_{t+1} = i, i_t, \ldots, i_1, o_{t+1}, \ldots, o_1 \mid \lambda) \\
&= \max_{i_1,\ldots,i_t} P(i_{t+1} = i, o_{t+1} \mid i_t, \ldots, i_1, o_t, \ldots, o_1, \lambda)P(i_t, \ldots, i_1, o_t, \ldots, o_1 \mid \lambda) \\
&= \max_{i_1,\ldots,i_t} P(o_{t+1} \mid i_{t+1} = i, i_t, \ldots, i_1, o_t, \ldots, o_1, \lambda)P(i_{t+1} = i \mid i_t, \ldots, i_1, o_t, \ldots, o_1, \lambda) \\
&\quad P(i_t, \ldots, i_1, o_t, \ldots, o_1 \mid \lambda) \\
&= \max_{i_1,\ldots,i_t} P(o_{t+1} \mid i_{t+1} = i, \lambda)P(i_{t+1} = i \mid i_t, \lambda)P(i_t, \ldots, i_1, o_t, \ldots, o_1 \mid \lambda) \\
&= \max_{1\le j\le N} \max_{i_1,\ldots,i_{t-1}} P(o_{t+1} \mid i_{t+1} = i, \lambda)P(i_{t+1} = i \mid i_t = j, \lambda)P(i_t = j, \ldots, i_1, o_t, \ldots, o_1 \mid \lambda) \\
&= \max_{1\le j\le N} b_i(o_{t+1})a_{ji}\delta_t(j) = \max_{1\le j\le N} \left[\delta_t(j)a_{ji}\right] b_i(o_{t+1}), \quad i = 1, 2, \ldots, N,\ t = 1, 2, \ldots, T-1.
\end{aligned}$$

$\Psi_t(i) = \arg\max_{1\le j\le N} \left[\delta_{t-1}(j) \cdot a_{ji}\right], \quad i = 1, 2, \ldots, N$

3）终止：

$$P^* = \max_{1\le i\le N} \delta_T(i)$$
$$i_T^* = \arg\max_{1\le i\le N} \delta_T(i)$$

4）最优路径回溯，对 $t = T-1, T-2, \ldots, 1$

$$i_t^* = \Psi_{t+1}(i_{t+1}^*)$$

得到最优路径 $I^* = (i_1^*, i_2^*, \ldots, i_T^*)$

如下是在前向算法例题基础上，关于维特比算法的计算分析：

例. 考虑盒子和球模型 $\lambda = (A, B, \pi)$，状态集合 $Q = \{1, 2, 3\}$，观测集合 $V = \{红, 白\}$，

$$A = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5 & 0.5 \\ 0.4 & 0.6 \\ 0.7 & 0.3 \end{bmatrix}, \quad \pi = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \end{bmatrix}$$

设 $T = 3$，$O = (红, 白, 红)$，试求最优状态序列，即最优路径 $I^* = (i_1^*, i_2^*, \ldots, i_T^*)$。

1) 初始化：$\delta_1(i)=\pi_i b_i(o_1)$ ∴ $\delta_1(1)=0.10$，$\delta_1(2)=0.16$，$\delta_1(3)=0.28$

2). $\delta_2(i)=\max\limits_{1\le j\le 3}[\delta_1(j)a_{ji}]b_i(o_2)$，对每个状态 i 记录概率最大路径的前一个状态 j.

有：$\Psi_2(i)=\arg\max\limits_{1\le j\le 3}[\delta_1(j)a_{ji}]$，$i=1,2,3$.

∴ $\delta_2(1)=\max\limits_{1\le j\le 3}[\delta_1(j)a_{j1}]b_1(o_2)=0.028$，$\Psi_2(1)=3$；$\delta_2(2)=0.0504$，$\Psi_2(2)=3$

$\delta_2(3)=0.042$，$\Psi_2(3)=3$.

$t=3$ 时，$\delta_3(1)=0.00756$，$\Psi_3(1)=2$；$\delta_3(2)=0.01008$，$\Psi_3(2)=2$；$\delta_3(3)=0.0147$，$\Psi_3(3)=3$.

可以构造下图：

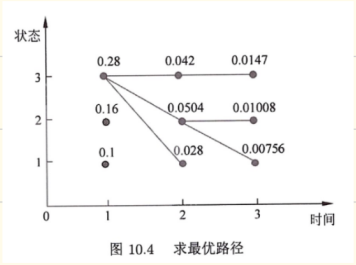


图 10.4 求最优路径

3) 最优路径概率：$P^*=\max\limits_{1\le i\le 3}\delta_3(i)=0.0147$；终点为 $i_3^*=\arg\max\limits_i \delta_3(i)=3$.

4). 逆向找到 i_2^*，i_1^*：$i_2^*=\Psi_3(i_3^*)=3$；$i_1^*=\Psi_2(i_2^*)=3$. ∴ $I^*=(3,3,3)$.

9.4 学习问题

给定模型观测数据：$O=(o_1,o_2,\ldots,o_T)$，利用最大似然 $P(O\mid\lambda)$ 计算参数模型：$\lambda=(A,B,\pi)$

- **监督学习方法**：假设训练数据是包括观测序列 O 和对应的状态序列 I——$\{(O_1,I_1),(O_2,I_2),\ldots,(O_S,I_S)\}$，可以直接利用**极大似然估计法**来估计隐马尔可夫模型参数。

  1）转移概率 a_{ij} 的估计：时刻 t 处于状态 i 而时刻 $t+1$ 转移到状态 j 的频数为 A_{ij}，则 a_{ij} 的估计为

  $$\hat{a}_{ij}=\frac{A_{ij}}{\sum_{j=1}^{N}A_{ij}},\quad i=1,2,\ldots,N,\quad j=1,2,\ldots,N$$

  2）观测概率 $b_j(k)$ 的估计：样本中状态为 j 并观测为 k 的频数是 B_{jk}，那么状态为 j、观测为 k 的概率 $b_j(k)$ 估计是

  $$\hat{b}_j(k)=\frac{B_{jk}}{\sum_{k=1}^{M}B_{jk}},\quad j=1,2,\ldots,N,\quad k=1,2,\ldots,M$$

  3）初始状态概率 π_i 的估计 $\hat{\pi}_i$ 为 S 个样本中初始状态为 q_i 的频率。

- 非监督学习方法：假设训练数据只有观测序列 $O=(o_1,o_2,\ldots,o_T)$，采用 Baum-Welch 算法（EM 算法的特殊情况）

10 无监督学习——聚类

回顾先前所学，我们都是在已知标签的情况下处理数据，属于监督学习，而此处的非监督学习是没有标签的数据直接进行建模。如下可对概念进行简单辨析：

- **监督学习**：依赖于**标记数据**的模型，训练数据是一套训练样本，每个样本都是由一个**输入对象**和一个**期望的输出值**组成。常用于分类和回归。
- **半监督学习**：介于监督学习与无监督学习相结合的一种学习方法。利用**少量的标注样本**和**大量的未标注样本**进行训练和分类的问题。
- **无监督学习**：没有任何标记的训练样本，直接对数据进行建模，寻找数据的内在结构及规律，如类别和聚类。常用于聚类、降维、概率密度估计。

10.1 类（簇）、距离

聚类得到的“类”或“簇”，就是**样本的子集**。**硬聚类**——一个样本只能属于一个类，类与类之间没有交集（如 K-Means 聚类）；**软聚类**——一个样本可以同时属于多个类，每个类的归属用概率 / 权重表示。

10.1.1 类/簇

用 G 表示类或簇（cluster），x_i 表示类中的样本，n_G 表示 G 中样本的个数，d_{ij} 表示样本 x_i 与样本 x_j 之间的距离。如下是两种对类/簇的定义：

- 设 T 为给定的正数，若集合 G 中任意两个样本 x_i, x_j，有 $d_{ij} \leq T$，则称 G 为一个类或簇。
- 设 T 为给定的正数，若对集合 G 的任意样本 x_i，一定存在 G 中的另一个样本 x_j，使得 $d_{ij} \leq T$，则称 G 为一个类或簇。

类与簇的特征可以通过不同角度来刻画，常用的特征有下面三种：

1. **类的中心**：即类的均值

$$\bar{x}_G = \frac{1}{n_G} \sum_{i=1}^{n_G} x_i$$

2. **类的直径** D_G：类中任意两个样本之间的最大距离，即

$$D_G = \max_{x_i, x_j \in G} d_{ij}$$

3. **类的样本散布矩阵** A_G

$$\sum_{i=1}^{n_G}(x_i-\bar{x}_G)(x_i-\bar{x}_G)^{\mathrm{T}}$$

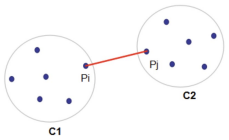
4. **样本协方差**矩阵 S_G

$$S_G=\frac{1}{n_G-1}A_G=\frac{1}{n_G-1}\sum_{i=1}^{n_G}(x_i-\bar{x}_G)(x_i-\bar{x}_G)^{\mathrm{T}}$$

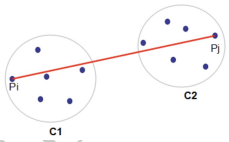
10.1.2 距离的定义

两个类 G_p 与 G_q 之间的距离为 $D(p,q)$，也称为**连接**。G_p 类中包含 n_p 个样本，G_q 类中包含 n_q 个样本，分别 $\bar{x_p}$ 和 $\bar{x_q}$ 表示 G_p 和 G_q 的均值，即类的中心。

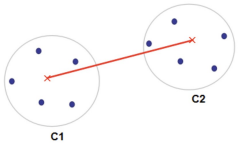
- 最短距离或单连接：$D_{pq}=\min\{d_{ij}\mid x_i\in G_p,x_j\in G_q\}$



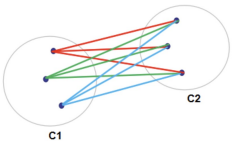
- 最长距离或完全连接：$D_{pq}=\max\{d_{ij}\mid x_i\in G_p,x_j\in G_q\}$



- 中心距离：$D_{pq}=d_{\bar{x}_p\bar{x}_q}$



- 平均距离：$D_{pq}=\frac{1}{n_pn_q}\sum_{x_i\in G_p}\sum_{x_j\in G_q}d_{ij}$



相似度：假设有 n 个样本，每个样本由 m 个属性的特征向量组成，样本合集可以用矩阵 X 表示

$$X=[x_{ij}]_{m\times n}=\begin{bmatrix}x_{11}&x_{12}&\cdots&x_{1n}\\x_{21}&x_{22}&\cdots&x_{2n}\\\vdots&\vdots&&\vdots\\x_{m1}&x_{m2}&\cdots&x_{mn}\end{bmatrix}$$

聚类的核心概念是**相似度**或**距离**，有多种相似度或距离定义，相似度直接影响聚类的结果，其选择是聚类的根本问题。

10.1.3 距离

- **闵可夫斯基距离**：在 k 近邻部分便已经提及过该距离，此处简单回顾。闵可夫斯基距离越大，相似度越小。给定样本集合 X, X 是 m 维实数向量空间 R^m 中点的集合，其中 $x_i, x_j \in X$, $x_i = (x_{1i}, x_{2i}, \ldots, x_{mi})^{\mathrm{T}}$, $x_j = (x_{1j}, x_{2j}, \ldots, x_{mj})^{\mathrm{T}}$ 样本 x_i 与样本 x_j 的闵可夫斯基距离定义为

$$d_{ij} = \left(\sum_{k=1}^{m} |x_{ki} - x_{kj}|^p \right)^{\frac{1}{p}}, \ p \geq 1$$

当 $p = 1$ 时为曼哈顿距离，$p = 2$ 时为欧氏距离，$p = \infty$ 时为切比雪夫距离。

- **马哈拉诺比斯距离**：简称马氏距离，也是另一种常用的相似度，考虑各个分量（特征）之间的相关性并与各个分量的尺度无关。距离越大，相似度越小。给定一个样本集合 X, $X = [x_{ij}]_{m \times n}$，其协方差矩阵记作 S。样本 x_i 与样本 x_j 之间的马哈拉诺比斯距离 d_{ij} 定义为

$$d_{ij} = \left[(x_i - x_j)^{\mathrm{T}} S^{-1} (x_i - x_j) \right]^{\frac{1}{2}}$$

$$x_i = (x_{1i}, x_{2i}, \ldots, x_{mi})^{\mathrm{T}}, \quad x_j = (x_{1j}, x_{2j}, \ldots, x_{mj})^{\mathrm{T}}$$

- **相关系数**：样本间的相似度用相关系数表示，相关系数的绝对值越接近于 1，表示样本越相似，越接近于 0，表示样本越不相似。样本 x_i 与样本 x_j 之间的相关系数定义为

$$r_{ij} = \frac{\sum_{k=1}^{m} (x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j)}{\left[\sum_{k=1}^{m} (x_{ki} - \bar{x}_i)^2 \sum_{k=1}^{m} (x_{kj} - \bar{x}_j)^2 \right]^{\frac{1}{2}}}, \quad \bar{x}_i = \frac{1}{m} \sum_{k=1}^{m} x_{ki}, \quad \bar{x}_j = \frac{1}{m} \sum_{k=1}^{m} x_{kj}$$

- **夹角余弦**：夹角余弦越接近于 1，表示样本越相似，越接近于 0，表示样本越不相似。样本 x_i 与样本 x_j 之间的夹角余弦定义为样本 x_i 与样本 x_j 之间的夹角余弦定义为

$$s_{ij} = \frac{\sum_{k=1}^{m} x_{ki} x_{kj}}{\left[\sum_{k=1}^{m} x_{ki}^2 \sum_{k=1}^{m} x_{kj}^2 \right]^{\frac{1}{2}}}$$

10.2 层次聚类

假定样本集 $D = \{x_1, x_2, \ldots, x_m\}$ 包含 m 个无标记样本，每个样本 $x_i = (x_{i1}; x_{i2}; \ldots; x_{in})$ 是一个 n 维的特征向量，聚类算法将样本集 D 划分成 k 个不相交的簇 $\{C_l | l = 1, 2, \ldots, k\}$，其中 $D = \bigcup_{l=1}^{k} C_l$ 且 $C_{l'} \cap \bigcap_{l' \neq l} C_l = \phi$。相应地，用 $\lambda_j \in \{1, 2, \ldots, k\}$ 表示样本 x_j 的 “簇标记”，即 $x_j \in C_{\lambda_j}$。于是，聚类的结果可用包含 m 个元素的**簇标记向量** $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$

层次聚类假设了类别之间存在层次结构，将样本聚到层次化的类中，每个样本只属于一个类，属于**硬聚类**。层次聚类三要素：

- **距离或相似度**：闵可夫斯基距离、马哈拉诺比斯距离、相关系数、夹角余弦

- **合并规则**：类间距离最小，类间距离的选择多样，如最短距离、最长距离、中心距离、平均距离
- **停止条件**：类的个数达到闭值，或直径超过阈值

层次聚类有**两种方法**—— 聚合聚类 （自下而上聚类）和 分裂聚类 （自上而下聚类）

- **聚合聚类**：开始将每个样本各自分到一个类；将相距最近的两类合并，建立一个新的类；重复此操作直到满足停止条件；得到层次化的类别。
- **分裂聚类**：开始将所有样本分到一个类；将已有类中相距最远的样本分到两个新的类；重复此操作直到满足停止条件；得到层次化的类别。

如下是聚合聚类的计算示例：

5 个样本的集合，样本之间的欧氏距离由如下矩阵 D 表示为

$$D=[d_{ij}]_{5\times 5}=\begin{bmatrix}0&7&2&9&3\\7&0&5&4&6\\2&5&0&8&1\\9&4&8&0&5\\3&6&1&5&0\end{bmatrix}$$

1. 由矩阵 D 可以看出，$d_{35}=d_{53}=1$ 为最小，所以把 G_3 和 G_5 合并为一个新类，记作 $G_6=\{x_3,x_5\}$。
2. 计算 G_6 与 G_1,G_2,G_4 之间的最短距离，以及其余两类之间的距离，得到结果将 G_1 与 G_6 合并成一个新类，记作 $G_7=\{x_1,x_3,x_5\}$。
3. 再计算 G_7 与 G_2,G_4 之间的最短距离，以及比较 G_2 与 G_4 间的距离。最后选取将 G_2 与 G_4 合并成一个新类，记作 $G_8=\{x_2,x_4\}$。
4. 将 G_7 与 G_8 合并成一个新类，记作 $G_9=\{x_1,x_2,x_3,x_4,x_5\}$，即将全部样本聚成 1 类，聚类终止。

10.3 k 均值聚类

k 均值聚类是基于样本集合划分的聚类算法，将样本集合划分为 k 个子集，构成 k 个类，将 n 个样本分到 k 个类中，每个样本到其所属类的中心的距离最小。每个样本只能属于一个类，所以 k 均值聚类是硬聚类。

样本的集合 $X=\{x_1,x_2,\ldots,x_n\}$,每个样本都是 m 维向量,将这 n 个样本分到 k 个类 $G_1,G_2,\ldots,G_k$ 中，其中

$$G_i\cap G_j=\varnothing,\ \bigcup_{i=1}^{k}G_i=X$$

用 C 表示划分，可以用函数 $l=C(i)$ 表示，其中样本用一个整数 $i\in\{1,2,\ldots,n\}$ 表示，类用一个整数 $l\in\{1,2,\ldots,k\}$ 表示。

评价标准：采用欧氏距离**平方**作为样本之间的距离 $d(x_i, x_j)$

$$d(x_i, x_j) = \sum_{k=1}^{m} (x_{ki} - x_{kj})^2 = \|x_i - x_j\|^2$$

定义样本与其所属类的中心之间的距离的总和为损失函数：

$$W(C) = \sum_{l=1}^{k} \sum_{C(i)=l} \|x_i - \bar{x}_l\|^2$$

由此 k 均值聚类就是求解损失函数最小的最优化问题。

训练策略

- **初始化**：随机选择 k 个样本作为初始类中心 $m_1, m_2, \ldots, m_k$；
- **分配样本**：给定当前类中心，求解目标函数 $\min_C \sum_{l=1}^{k} \sum_{C(i)=l} \|x_i - m_l\|^2$，将每个样本 x_i 指派到与其最近的中心 m_i 对应的类 G_l 中，得到新的划分 C；
- **中心更新**：给定当前划分 C，求解目标函数 $\min_{m_1,\ldots,m_k} \sum_{l=1}^{k} \sum_{C(i)=l} \|x_i - m_l\|^2$，对每个类 G_l **更新中心为类内样本均值**

$$m_l = \frac{1}{n_l} \sum_{C(i)=l} x_i, \quad l = 1, \ldots, k$$

- 重复样本分配和中心更新步骤，直到划分 C 不再变化（或中心变化小于阈值），算法终止，得到最终的聚类结果。

k 均值聚类算法**复杂度**是 $O(mnk)$，其中 m 是样本维数，n 是样本个数，k 是类别个数。

如下是 k 均值聚类的计算 示例：

给定含有 5 个样本的集合，试用 k 均值聚类算法将样本聚到 2 个类中。

$$X = \begin{bmatrix} 0 & 0 & 1 & 5 & 5 \\ 2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

1. 选择两个样本点作为类的中心。假设选择：$m_1^{(0)} = x_1 = (0, 2)^{\mathrm{T}}$，$m_2^{(0)} = x_2 = (0, 0)^{\mathrm{T}}$；以 $m_1^{(0)}$ 为类 $G_1^{(0)}$，$m_2^{(0)}$ 为类 $G_2^{(0)}$ 的中心。

2. 计算样本到类的距离并分配样本：

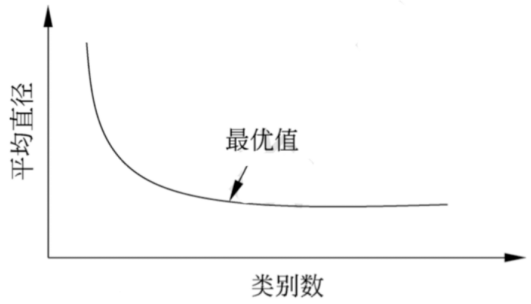
   - 对 $x_3 = (1, 0)^{\mathrm{T}}$：$d(x_3, m_1^{(0)}) = 5$, $d(x_3, m_2^{(0)}) = 1$, 将 x_3 分到类 $G_2^{(0)}$。
   - 对 $x_4 = (5, 0)^{\mathrm{T}}$：$d(x_4, m_1^{(0)}) = 29$, $d(x_4, m_2^{(0)}) = 25$, 将 x_4 分到类 $G_2^{(0)}$。
   - 对 $x_5 = (5, 2)^{\mathrm{T}}$：$d(x_5, m_1^{(0)}) = 25$, $d(x_5, m_2^{(0)}) = 29$, 将 x_5 分到类 $G_1^{(0)}$。

3. 得到新的类：$G_1^{(1)} = \{x_1, x_5\}$，$G_2^{(1)} = \{x_2, x_3, x_4\}$；中心为：$m_1^{(1)} = (2.5, 2.0)^{\mathrm{T}}$，$m_2^{(1)} = (2, 0)^{\mathrm{T}}$

4. 重复步骤，得到新的类发现没有改变，因此聚类停止，得到聚类结果 $G_1^* = \{x_1, x_5\}$，$G_2^* = \{x_2, x_3, x_4\}$。

算法特性：k 均值聚类是基于事先划分类别数 k 的聚类，得到的类别是平坦的、非层次化的。且算法是迭代算法，**不能保证得到全局最优**，初始中心的选择会直接影响聚类结果（虽然类中心在聚类的过程中会发生移动，但是往往不会移动太大，因为在每一步，样本都已经被分到与其最近的中心的类中）。

实际应用中最优的 k 值是不知道的，因此需要尝试使用不同的 k 值聚类，检验得到聚类结果的质量，推测最优的 k 值。**聚类结果的质量可以用类的平均直径来衡量**。当类别数变大超过某个值以后，**平均直径不变**，而这个值正是最优的 k 值。



11 无监督学习——降维

11.1 主成分分析的主要思想

主成分分析（PCA）是一种常用的无监督学习方法，其利用**正交变换**把由线性相关变量表示的观测数据转换为少数几个由**线性无关变量**表示的数据，线性无关的变量称为**主成分**（个数通常小于原始变量的个数，达到降维的效果）。PCA 主要用于发现数据中变量之间的关系。

总体主成分分析——数据总体上进行的主成分分析；**样本主成分分析**——有限样本上进行的主成分分析。总体主成分分析是样本主成分分析的基础。

PCA 核心逻辑：1）**数据预处理**——标准化（均值为 0，方差为 1），消除量纲影响。如对随机变量 x_i 规范化为 $x_i^* = \frac{x_i - E(x_i)}{\sqrt{\mathrm{var}(x_i)}}, \quad i = 1, 2, \ldots, m$；2）**找最优方向**——第一主成分：方差最大的正交方向（信息最多）；后续主成分：与前面正交、方差次大的方向（信息递减）；3）**降维逻辑**——投影到低维度，新的低维合成特征替代原始高维特征。

11.2 总体主成分分析

前置内容：

有 m 维随机变量 $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)^{\mathrm{T}}$，其均值向量为 $\mu = E(\boldsymbol{x}) = (\mu_1, \mu_2, \ldots, \mu_m)^{\mathrm{T}}$；协方差矩阵是 $\Sigma = \mathrm{cov}(\boldsymbol{x}, \boldsymbol{x}) = E\left[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^{\mathrm{T}}\right]$，考虑 x 到 m 维随机变量 $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)^{\mathrm{T}}$ 的线性变换为 $y_i = \boldsymbol{\alpha}_i^{\mathrm{T}}\boldsymbol{x} = \alpha_{1i}x_1 + \alpha_{2i}x_2 + \cdots + \alpha_{mi}x_m$，其中 $\boldsymbol{\alpha}_i^{\mathrm{T}} = (\alpha_{1i}, \alpha_{2i}, \ldots, \alpha_{mi}), \quad i = 1, 2, \ldots, m$。

性质：

$$
\begin{aligned}
E(y_i) &= \boldsymbol{\alpha}_i^{\mathrm{T}}\boldsymbol{\mu}, && i = 1, 2, \ldots, m \\
\mathrm{var}(y_i) &= \boldsymbol{\alpha}_i^{\mathrm{T}}\Sigma\boldsymbol{\alpha}_i, && i = 1, 2, \ldots, m \\
\mathrm{cov}(y_i, y_j) &= \boldsymbol{\alpha}_i^{\mathrm{T}}\Sigma\boldsymbol{\alpha}_j, && i = 1, 2, \ldots, m; \quad j = 1, 2, \ldots, m
\end{aligned}
$$

总体主成分分析满足三个条件：

- 系数向量 $\boldsymbol{\alpha}_i$ 是单位向量，即 $\boldsymbol{\alpha}_i^{\mathrm{T}}\boldsymbol{\alpha}_i = 1$，$i = 1, 2, \ldots, m$；
- 变量 y_i 与 y_j 互不相关，即 $\mathrm{cov}(y_i, y_j) = 0$ $(i \neq j)$；
- 变量 y_1 是 $\boldsymbol{x}$ 的所有线性变换中方差最大的；y_2 是与 y_1 不相关的 $\boldsymbol{x}$ 的所有线性变换中方差最大的；一般地，y_i 是与 $y_1, y_2, \ldots, y_{i-1}$ $(i = 1, 2, \ldots, m)$ 都不相关的 $\boldsymbol{x}$ 的所有线性变换中方差最大的；这时分别称 $y_1, y_2, \ldots, y_m$ 为 $\boldsymbol{x}$ 的第一主成分、第二主成分、…、第 m 主成分。

对于第一个条件有：

$$
\boldsymbol{\alpha}_i^{\mathrm{T}}\boldsymbol{\alpha}_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}
$$

第二、三个条件即为：在 $\boldsymbol{\alpha}_k^{\mathrm{T}}\boldsymbol{\alpha}_k = 1$ 条件下，同时在与 $\boldsymbol{\alpha}_1^{\mathrm{T}}\boldsymbol{x}, \ldots, \boldsymbol{\alpha}_{k-1}^{\mathrm{T}}\boldsymbol{x}$ 不相关的 x 的所有线性变换 $\boldsymbol{\alpha}_k^{\mathrm{T}}\boldsymbol{x} = \sum_{i=1}^{m}\alpha_{ik}x_i$ 中，求方差最大的，得到 x 的第 k 主成分。

计算第一主成分：求解约束最优化问题

$$\begin{aligned}\max_{\boldsymbol{\alpha}_1} \quad & \boldsymbol{\alpha}_1^{\mathrm{T}}\Sigma\boldsymbol{\alpha}_1 \\ \text{s.t.} \quad & \boldsymbol{\alpha}_1^{\mathrm{T}}\boldsymbol{\alpha}_1 = 1\end{aligned}$$

拉格朗日函数为 $\boldsymbol{\alpha}_1^{\mathrm{T}}\Sigma\boldsymbol{\alpha}_1 - \lambda_1\left(\boldsymbol{\alpha}_1^{\mathrm{T}}\boldsymbol{\alpha}_1 - 1\right)$，对 $\boldsymbol{\alpha}_1$ 求偏导（公式：$\frac{\partial \boldsymbol{x}^{\mathrm{T}}A\boldsymbol{x}}{\partial \boldsymbol{x}} = \left(A + A^{\mathrm{T}}\right)\boldsymbol{x}$ （A 为方阵））并等于 0，有

$$\Sigma\boldsymbol{\alpha}_1 - \lambda_1\boldsymbol{\alpha}_1 = 0$$

得到第一主成分为 $\operatorname{var}\left(\boldsymbol{\alpha}_1^{\mathrm{T}}\boldsymbol{x}\right) = \boldsymbol{\alpha}_1^{\mathrm{T}}\Sigma\boldsymbol{\alpha}_1 = \boldsymbol{\alpha}_1^{\mathrm{T}}\lambda_1\boldsymbol{\alpha}_1 = \lambda_1\boldsymbol{\alpha}_1^{\mathrm{T}}\boldsymbol{\alpha}_1 = \lambda_1$

计算第二主成分：与上述过程相似，依旧是求解约束最优化问题

$$\begin{aligned}\max_{\boldsymbol{\alpha}_2} \quad & \boldsymbol{\alpha}_2^{\mathrm{T}}\Sigma\boldsymbol{\alpha}_2 \\ \text{s.t.} \quad & \boldsymbol{\alpha}_1^{\mathrm{T}}\Sigma\boldsymbol{\alpha}_2 = 0, \quad \boldsymbol{\alpha}_2^{\mathrm{T}}\Sigma\boldsymbol{\alpha}_1 = 0 \\ & \boldsymbol{\alpha}_2^{\mathrm{T}}\boldsymbol{\alpha}_2 = 1\end{aligned}$$

由于第二主成分与前面的主成分不相关，因此约束条件多了 $\boldsymbol{\alpha}_1^{\mathrm{T}}\Sigma\boldsymbol{\alpha}_2 = 0, \quad \boldsymbol{\alpha}_2^{\mathrm{T}}\Sigma\boldsymbol{\alpha}_1 = 0$ 部分，协方差为 0，两个主成分间不相关。前面的主成分约束进一步化简：

$$\begin{aligned}&\boldsymbol{\alpha}_1^{\mathrm{T}}\Sigma\boldsymbol{\alpha}_2 = \boldsymbol{\alpha}_2^{\mathrm{T}}\Sigma\boldsymbol{\alpha}_1 = \boldsymbol{\alpha}_2^{\mathrm{T}}\lambda_1\boldsymbol{\alpha}_1 = \lambda_1\boldsymbol{\alpha}_2^{\mathrm{T}}\boldsymbol{\alpha}_1 = \lambda_1\boldsymbol{\alpha}_1^{\mathrm{T}}\boldsymbol{\alpha}_2 = 0 \\ &\Rightarrow \boldsymbol{\alpha}_1^{\mathrm{T}}\boldsymbol{\alpha}_2 = \boldsymbol{\alpha}_2^{\mathrm{T}}\boldsymbol{\alpha}_1 = 0\end{aligned}$$

由此得到拉格朗日函数为 $\boldsymbol{\alpha}_2^{\mathrm{T}}\Sigma\boldsymbol{\alpha}_2 - \lambda\left(\boldsymbol{\alpha}_2^{\mathrm{T}}\boldsymbol{\alpha}_2 - 1\right) - \phi\boldsymbol{\alpha}_2^{\mathrm{T}}\boldsymbol{\alpha}_1$；最后解得第二主成分

$$\boldsymbol{\alpha}_2^{\mathrm{T}}\Sigma\boldsymbol{\alpha}_2 = \boldsymbol{\alpha}_2^{\mathrm{T}}\lambda_2\boldsymbol{\alpha}_2 = \lambda_2\boldsymbol{\alpha}_2^{\mathrm{T}}\boldsymbol{\alpha}_2 = \lambda_2$$

按照上述方法求得第一、第二、直到第 m 主成分，其系数向量 $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_m$ 分别是 Σ 的第一个、第二个、直到第 m 个单位特征向量，$\lambda_1, \lambda_2, \ldots, \lambda_m$ 分别是对应的特征值，且有 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$。第 k 主成分的方差等于 Σ 的第 k 个特征值，

$$\operatorname{var}(\boldsymbol{\alpha}_k^{\mathrm{T}}\boldsymbol{x}) = \boldsymbol{\alpha}_k^{\mathrm{T}}\Sigma\boldsymbol{\alpha}_k = \lambda_k, \quad k = 1, 2, \ldots, m$$

矩阵形式：随机变量 $y = (y_1, y_2, \ldots, y_m)^{\mathrm{T}}$ 的分量依次是 x 的第一主成分到第 m 主成分的充要条件为 1）$y = A^{\mathrm{T}}x$，A 为正交矩阵

$$A=\begin{bmatrix}\alpha_{11} & \alpha_{12} & \cdots & \alpha_{1m}\\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2m}\\ \vdots & \vdots & & \vdots\\ \alpha_{m1} & \alpha_{m2} & \cdots & \alpha_{mm}\end{bmatrix}$$

2）y 的协方差矩阵为对角矩阵

$$\mathrm{cov}(\boldsymbol{y})=\mathrm{diag}(\lambda_1,\lambda_2,\ldots,\lambda_m)$$

$$\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_m$$

其中，λ_k 是 Σ 的第 k 个特征值，α_k 是对应的单位特征向量，$k=1,2,\ldots,m$。

性质：

- 总体主成分 y 的协方差矩阵是对角矩阵：$\mathrm{cov}(\boldsymbol{y})=\Lambda=\mathrm{diag}(\lambda_1,\lambda_2,\ldots,\lambda_m)$
- 总体主成分 y 的方差之和等于随机变量 x 的方差之和，即

  $$\sum_{i=1}^{m}\lambda_i=\sum_{i=1}^{m}\sigma_{ii}$$

  其中 σ_{ii} 是随机变量 x_i 的方差，即协方差矩阵 Σ 的对角线元素。

  $$\Sigma A=A\Lambda\longrightarrow A^T\Sigma A=A^TA\Lambda\xrightarrow{A\text{ 正交矩阵:}A^T=A^{-1}}A^T\Sigma A=\Lambda\quad\Rightarrow\quad\Sigma=A\Lambda A^T$$

  $$\sum_{i=1}^{m}\mathrm{var}(x_i)=\mathrm{tr}(\Sigma)=\mathrm{tr}(A\Lambda A^T)\xrightarrow{\mathrm{tr}(BC)=\mathrm{tr}(CB)}\mathrm{tr}(\Lambda A^TA)=\mathrm{tr}(\Lambda)=\sum_{i=1}^{m}\lambda_i=\sum_{i=1}^{m}\mathrm{var}(y_i)$$

- 第 k 个主成分 y_k 与变量 x_i 的相关系数 $\rho(y_k,x_i)$ 称为**因子负荷量**。

  $$\mathrm{cov}(y_k,x_i)=\mathrm{cov}(\alpha_k^Tx,e_i^Tx)=\alpha_k^T\Sigma e_i=e_i^T\Sigma\alpha_k=e_i^T\lambda_k\alpha_k=\lambda_k\alpha_{ik}$$

  由此：

  $$\rho(y_k,x_i)=\frac{\mathrm{cov}(y_k,x_i)}{\sqrt{\mathrm{var}(y_k)\mathrm{var}(x_i)}}=\frac{\mathrm{cov}(\alpha_k^Tx,e_i^Tx)}{\sqrt{\lambda_k}\sqrt{\sigma_{ii}}}=\frac{\sqrt{\lambda_k}\alpha_{ik}}{\sqrt{\sigma_{ii}}},\quad k,i=1,2,\ldots,m$$

  共 m 个主成分时，第 m 个主成分与第 i 个变量的因子负荷量满足 $\sum_{k=1}^{m}\rho^2(y_k,x_i)=1$

  推导：

  $$\sigma_{ii}=\mathrm{cov}(X_i)\xrightarrow{\text{协方差矩阵}\Sigma\text{的}i\text{行}i\text{列}}$$

$$\Sigma = A\Lambda A^T = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1m} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m1} & \alpha_{m2} & \cdots & \alpha_{mm} \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_m \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{21} & \cdots & \alpha_{m1} \\ \alpha_{12} & \alpha_{22} & \cdots & \alpha_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{1m} & \alpha_{2m} & \cdots & \alpha_{mm} \end{bmatrix}$$

$$= \begin{bmatrix} \lambda_1\alpha_{11} & \lambda_2\alpha_{12} & \cdots & \lambda_m\alpha_{1m} \\ \lambda_1\alpha_{21} & \lambda_2\alpha_{22} & \cdots & \lambda_m\alpha_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1\alpha_{m1} & \lambda_2\alpha_{m2} & \cdots & \lambda_m\alpha_{mm} \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{21} & \cdots & \alpha_{m1} \\ \alpha_{12} & \alpha_{22} & \cdots & \alpha_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{1m} & \alpha_{2m} & \cdots & \alpha_{mm} \end{bmatrix}$$

$$\text{第}i\text{行}i\text{列}: \begin{bmatrix} \lambda_1\alpha_{i1} & \lambda_2\alpha_{i2} & \cdots & \lambda_m\alpha_{im} \end{bmatrix} \begin{bmatrix} \alpha_{i1} \\ \alpha_{i2} \\ \vdots \\ \alpha_{im} \end{bmatrix} = \lambda_1\alpha_{i1}^2 + \lambda_2\alpha_{i2}^2 + \cdots + \lambda_m\alpha_{im}^2 = \sum_{k=1}^m \lambda_k\alpha_{ik}^2$$

$$\therefore \sum_{k=1}^m \rho^2(y_k, x_i) = \sum_{k=1}^m \frac{\lambda_k\alpha_{ik}^2}{\sigma_{ii}} \xlongequal{\sigma_{ii}=\sum_{k=1}^m \lambda_k\alpha_{ik}^2} \frac{\sum_{k=1}^m \lambda_k\alpha_{ik}^2}{\sum_{k=1}^m \lambda_k\alpha_{ik}^2} = 1$$

主成分的个数：通常取 k 使**累计方差贡献率**（反映了主成分保留信息的比例）达到规定的百分比以上，即 $\eta_k = \frac{\lambda_k}{\sum_{i=1}^m \lambda_i}$ 达到百分比即可。但是这个式子不能反映对变量 x_i 保留信息的比例，要讨论 k 个主成分对原有变量 x_i 的贡献率，将其定义为 x_i 与 $(y_1, y_2, \ldots, y_k)$ 的相关系数的平方，即为

$$\nu_i = \rho^2(x_i, (y_1, y_2, \ldots, y_k)) = \sum_{j=1}^k \rho^2(x_i, y_j) = \sum_{j=1}^k \frac{\lambda_j\alpha_{ij}^2}{\sigma_{ii}}$$

11.3 样本主成分分析

先前的总体 PCA 是针对 $\boldsymbol{x}$ 这个 m 维随机变量，但是在实际中没有随机变量，而是变量在 n 个样本上的观测值。我们在这些实际的**观测数据**上进行 PCA 就是样本主成分分析，它和总体主成分有相同性质。

对 m 维随机变量 $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)^{\mathrm{T}}$ 进行 n 次独立观测，有观测数据样本矩阵 X 为

$$X = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

注意：这里需要区分一下我们观测样本是个 m 维向量，观测样本为**黑体**的 $\boldsymbol{x_1}$ 这种，与随机变量 $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)^{\mathrm{T}}$ 中的 x_1 这种不是同一东西，后者代表的是特征 x_1。

- 样本均值为 $\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{j=1}^n \boldsymbol{x}_j$；
- 样本协方差矩阵为 $S = [s_{ij}]_{m\times m}$，　$s_{ij} = \frac{1}{n-1}\sum_{k=1}^n (x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j)$，　$i, j = 1, 2, \ldots, m$
- 样本相关矩阵 R 为 $R = [r_{ij}]_{m\times m}$，　$r_{ij} = \frac{s_{ij}}{\sqrt{s_{ii}s_{jj}}}$，　$i, j = 1, 2, \ldots, m$

- m 维向量 $\boldsymbol{x}$ 到 m 维向量 $\boldsymbol{y}=(y_1,y_2,\ldots,y_m)^T$ 的线性变换为 $\boldsymbol{y}=A^T\boldsymbol{x}$，其中

$$A=\begin{bmatrix}a_1 & a_2 & \cdots & a_m\end{bmatrix}=\begin{bmatrix}a_{11} & a_{12} & \cdots & a_{1m}\\ a_{21} & a_{22} & \cdots & a_{2m}\\ \vdots & \vdots & & \vdots\\ a_{m1} & a_{m2} & \cdots & a_{mm}\end{bmatrix}$$

$$a_i=(a_{1i},a_{2i},\ldots,a_{mi})^{\mathrm{T}},\quad i=1,2,\ldots,m$$

根据 $\boldsymbol{y_i}=a_i^{\mathrm{T}}\boldsymbol{x}=a_{1i}x_1+a_{2i}x_2+\cdots+a_{mi}x_m,\quad i=1,2,\ldots,m$ 可以推导得到

性质：

$\boldsymbol{y_i}$ 的样本均值：$\bar{\boldsymbol{y_i}}=\frac{1}{n}\sum_{j=1}^{n}a_i^{\mathrm{T}}\boldsymbol{x_j}=a_i^{\mathrm{T}}\bar{\boldsymbol{x}}$

$\boldsymbol{y_i}$ 的样本方差：$\mathrm{var}(\boldsymbol{y_i})=\frac{1}{n-1}\sum_{j=1}^{n}(a_i^{\mathrm{T}}\boldsymbol{x_j}-a_i^{\mathrm{T}}\bar{\boldsymbol{x}})^2=a_i^{\mathrm{T}}\left[\frac{1}{n-1}\sum_{j=1}^{n}(\boldsymbol{x_j}-\bar{\boldsymbol{x}})(\boldsymbol{x_j}-\bar{\boldsymbol{x}})^{\mathrm{T}}\right]a_i=a_i^{\mathrm{T}}Sa_i$

$\boldsymbol{y_i}$ 与 $\boldsymbol{y_j}$ 的样本协方差：$\mathrm{cov}(\boldsymbol{y_i},\boldsymbol{y_k})=a_i^{\mathrm{T}}Sa_k$

样本第一主成分是在 $a_1^Ta_1$ 条件下，使得 $a_1^{\mathrm{T}}\boldsymbol{x_j}$ $(j=1,2,\ldots,n)$ 的样本方差 $a_1^TSa_1$ 最大的 $\boldsymbol{x}$ 的线性变换；后续主成分也是在 $a_k^Ta_k=1$ 和与先前的主成分样本协方差 $a_k^TSa_i=0$ 的条件下，使得样本方差 $a_i^TSa_i$ 最大的 $\boldsymbol{x}$ 的线性变换。

计算过程：

1. 对观测数据进行规范化，得到的规范化矩阵仍记为 X：

   其中 $x_{ij}^*=\frac{x_{ij}-\bar{x}_i}{\sqrt{S_{ii}}},\quad i=1,2,\ldots,m;\quad j=1,2,\ldots,n$

2. **为了方便，后续将规范化后的变量 x_{ij}^* 仍记作 x_{ij}，规范化的样本矩阵仍记作 X**。因此，样本协方差矩阵 S 为样本相关矩阵 R：

$$R=[r_{ij}]_{m\times m}=\frac{1}{n-1}XX^{\mathrm{T}}$$

   其中

$$r_{ij}=\frac{1}{n-1}\sum_{l=1}^{n}x_{il}x_{lj},\quad i,j=1,2,\ldots,m$$

   样本协方差矩阵 S 为总体协方差矩阵 Σ 的无偏估计，其特征向量和特征值也是极大似然估计。

3. 求解 R 的特征方程 $|R-\lambda I|=0$ 得到特征值 $\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_m$，再求方差贡献率 $\sum_{i=1}^{k}\eta_i$ 达到预期值的主成分个数 k。再根据 $(R-\lambda I)a=0$ 和 $a^{\mathrm{T}}a=1$ 求得前 k 个特征值对应的单位特征向量 $a_i=(a_{1i},a_{2i},\ldots,a_{mi})^{\mathrm{T}},\quad i=1,2,\ldots,k$

4. 求 k 个主成分：$y_i=a_i^{\mathrm{T}}x,\quad i=1,2,\ldots,k$

5. 计算 k 个主成分 y_i 与原变量 x_i 的相关系数 $\rho(x_i,y_j)$，以及 k 个主成分对原变量 x_i 的贡献率 v_i

6. 计算 n 个样本的 k 个主成分值，即将样本投影在我们的各个主成分上。第 j 个样本 $x_j = (x_{1j}, x_{2j}, \ldots, x_{mj})^{\mathrm{T}}$ 的第 i 主成分值是

$$y_{ij} = (a_{1i}, a_{2i}, \ldots, a_{mi})(x_{1j}, x_{2j}, \ldots, x_{mj})^{\mathrm{T}} = \sum_{l=1}^{m} a_{li} x_{lj}, \quad i = 1, 2, \ldots, m, \quad j = 1, 2, \ldots, n$$

如例题：假设有 n 个学生参加四门课程的考试，将学生们的考试成绩看作随机变量的取值，对考试成绩数据进行标准化处理，得到样本相关矩阵 R。试对数据进行主成分分析。

课程	语文	外语	数学	物理
语文	1	0.44	0.29	0.33
外语	0.44	1	0.35	0.32
数学	0.29	0.35	1	0.60
物理	0.33	0.32	0.60	1

由题：样本相关矩阵 $R=\begin{pmatrix} 1 & 0.44 & 0.29 & 0.33 \\ 0.44 & 1 & 0.35 & 0.32 \\ 0.29 & 0.35 & 1 & 0.60 \\ 0.33 & 0.32 & 0.60 & 1 \end{pmatrix}$ x_1, x_2, x_3, x_4 分别表示语文、外语、数学、物理、

由 $|R-\lambda I|=0$ 与 $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4$ 得：$\lambda_1=2.17$；$\lambda_2=0.87$；$\lambda_3=0.57$；$\lambda_4=0.39$

选取累计方差贡献率大于75%，即 $k=2$ 只取前两个主成分。（因为 $\frac{\lambda_1+\lambda_2}{\sum_{i=1}^{4}\lambda_i}=0.76>0.75$）

再由 $\begin{cases}(R-\lambda_1 I)a_1=0 \\ (R-\lambda_2 I)a_2=0\end{cases}$；$a_1^{\mathrm{T}}a_1=0$；$a_2^{\mathrm{T}}a_2=0$ 依次解得单位特征向量：$a_1=(0.46, 0.476, 0.523, 0.537)^{\mathrm{T}}$

$a_2=(0.574, 0.486, -0.476, -0.456)^{\mathrm{T}}$

$\therefore y_1=0.460x_1+0.476x_2+0.523x_3+0.537x_4$

$y_2=0.574x_1+0.486x_2-0.476x_3-0.456x_4$

因子负荷量：$\rho(y_j, x_i)=\frac{\sqrt{\lambda_j}a_{ij}}{\sqrt{\sigma_{ii}}}=\sqrt{\lambda_j}a_{ij}$

贡献率：$V_i=\sum_{j=1}^{2}\rho^2(y_j, x_i)=\sum_{j=1}^{2}\lambda_j a_{ij}^2$

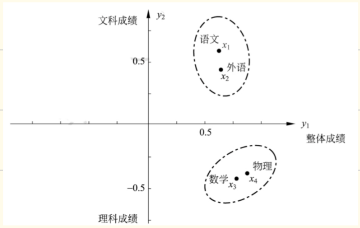
计算得：

项目	x_1	x_2	x_3	x_4
y_1	0.678	0.701	0.770	0.791
y_2	0.536	0.453	−0.444	−0.425
y_1, y_2 对 x_i 的贡献率	0.747	0.697	0.790	0.806

y_1 主成分因子负荷量均为正，表明 y_1 反映整体成绩且物理占最重要位置。

y_2 主成分反映了语文和外语、数学和物理的关系

两者表示为：



12 深度多层感知器和卷积神经网络

深度学习属于机器学习的一种，主要特点是使用**多层非线性处理单元**进行特征提取和转换。每层使用前一层的输出作为输入。与机器学习相比更强调“数据”和“层次化结构”。

12.1 深度 MLP 的使用技巧

在先前的多层感知机中，只要神经元数目足够多，双层感知器就能表达任意连续可微函数。但是网络结构的选择依赖于很多因素，当时是多层网络时，其面临着梯度消失和梯度爆炸问题、内部协变量偏移问题、过拟合问题这三个问题。

12.1.1 梯度消失和梯度爆炸问题

利用反向传播算法时，首先我们是正向传播计算神经网络各层的输出，然后通过反向传播计算神经网络各层的误差以及梯度，最后是梯度下降对各层参数进行更新。在这个过程中，各层的梯度，特别是前面的梯度，**有时会接近 0（梯度消失），或接近无穷（梯度爆炸）**。反向传播中的误差向量：

$$\delta^{(t-1)} = \left\{ \operatorname{diag}\left(\frac{\partial a}{\partial z^{(t-1)}} \right) \cdot W^{(t)\mathrm{T}} \right\} \cdot \delta^{(t)}$$

简化假设为 $\delta^{(t-1)} = \boldsymbol{U} \cdot \delta^{(t)}$，随着时间向前迭代得 $\delta^{(1)} = \boldsymbol{U}^{t-1} \cdot \delta^{(t)}$，$\boldsymbol{U}^{t-1}$ 收敛到 0 或发散到无穷。

防止梯度消失和梯度爆炸的**技巧**：

1）对每个神经元的参数权重初始值 $w = (w_1, w_2, \ldots, w_n)^{\mathrm{T}}$ 根据正态分布 $\mathcal{N}\left(0, \frac{1}{n}\right)$ 随机取值；

2）激活函数选择左饱和的整流线性函数，而不是双边饱和的 S 型函数；

3）使用特定的网络架构，如残差网络 (ResNet，CNN 的改进) 和 LSTM（RNN 的改进），避免反向传播时只依赖于矩阵连乘。

12.1.2 内部协变量偏移问题

在神经网络中，每一层的输出向量为 $\boldsymbol{h}^{(t)}$，其依赖于前一层的参数，并且作为“输入”又去训练下一层网络得到 $\boldsymbol{h}^{(t+1)}$。这就会使得在学习过程中，前面的参数在不断的迭代中更新，影响 $\boldsymbol{h}^{(t)}$ 的均值、方差、整体分布等，导致其动态变化，第 t 层及其后面层的学习不容易收敛。要解决这个问题则把每一层的输入样本进行归一化，有**批量归一化**（对每一层的净输入在每一个批量样本上进行归一化）和**层归一化**（对每一层的净输入在这一层上进行归一化）。

> 注意下面的输入!!! 批量归一化是用黑体 $\boldsymbol{z_i}$ 代表这是一个样本向量，这个样本含有多个维度；而层归一化是 z_i 代表一个样本输入的第 i 个神经元的输入。

批量归一化：假设批量数据在当前层的净输入是 $\{\boldsymbol{z_1}, \boldsymbol{z_2}, \ldots, \boldsymbol{z_n}\}$，$\boldsymbol{z_j}$ 是第 j 个样本的净输入，n 是批量样本数量。

- 计算当前层的净输入的均值与方差：$\boldsymbol{\mu} = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{z_j}$，$\boldsymbol{\sigma}^2 = \frac{1}{n-1}\sum_{j=1}^{n} \left(\boldsymbol{z_j} - \boldsymbol{\mu}\right)^2$
- 每一个样本的净输入进行归一化，得到向量：$\overline{z_j} = \frac{\boldsymbol{z_j} - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}}$，$j = 1, 2, \ldots, n$

- 进行仿射变换，$\boldsymbol{\gamma}$ 和 $\boldsymbol{\beta}$ 是参数向量（在所有批量中共有），$\odot$ 是向量的逐元素积（emmm，就是第一个元素之积 + 第二个元素之积 +……），归一化后仿射变换的结果作为批量数据这一层的净输入。

$$\tilde{\boldsymbol{z}}_j = \boldsymbol{\gamma} \odot \overline{\boldsymbol{z}}_j + \boldsymbol{\beta}, \quad j = 1, 2, \ldots, n$$

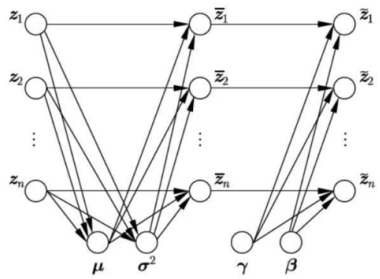
仿射变换把标准化后的数据重新缩放偏移到最合适的值。

根据上述批量归一化的训练过程，如下是**预测过程**的操作。

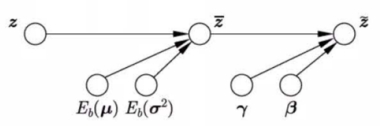
- 根据所有训练样本，计算所有批量的均值和方差的平均分别为 $E_b(\boldsymbol{\mu})$ 和 $E_b(\boldsymbol{\sigma})^2$；
- 将测试样本输入并在每一层的净输入进行归一化，得到归一化向量：

$$\overline{\boldsymbol{z}}_j = \frac{\boldsymbol{z}_j - E_b(\boldsymbol{\mu})}{\sqrt{E_b(\boldsymbol{\sigma}^2) + \epsilon}}, \quad j = 1, 2, \ldots, n$$

- 仿射变换：$\tilde{\boldsymbol{z}}_j = \boldsymbol{\gamma} \odot \overline{\boldsymbol{z}}_j + \boldsymbol{\beta}, \quad j = 1, 2, \ldots, n$，（这里 $\boldsymbol{\gamma}$ 和 $\boldsymbol{\beta}$ 是训练过程中得到的参数向量）



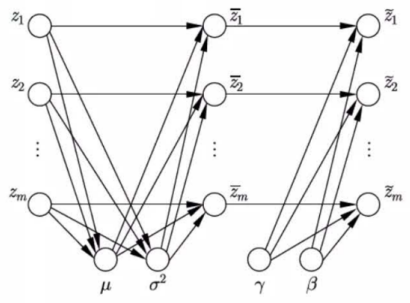
批量归一化：训练网络结构



批量归一化：推理网络结构

层归一化：和批量归一化类似，但是在**每一层**的神经元上进行归一化。当前层的神经元的净输入是 $z = (z_1, z_2, \ldots, z_m)^{\mathrm{T}}$，其中 z_j 是第 j 个神经元的净输入，m 是神经元个数。由于样本是单个 z，因此如下训练和预测过程操作一致。

- 计算这一层的神经元的净输入的均值与方差：$\mu = \frac{1}{m}\sum_{j=1}^{m} z_j, \quad \sigma^2 = \frac{1}{m-1}\sum_{j=1}^{m}\left(z_j - \mu\right)^2$
- 对每一个神经元的净输入进行归一化：$\overline{z}_j = \frac{z_j - \mu}{\sqrt{\sigma^2 + \epsilon}}, \quad j = 1, 2, \ldots, m$
- 进行仿射变换（γ 和 β 为参数）：$\tilde{z}_j = \gamma \cdot \overline{z}_j + \beta, \quad j = 1, 2, \ldots, m$，结果作为这一层神经元的实际净输入。

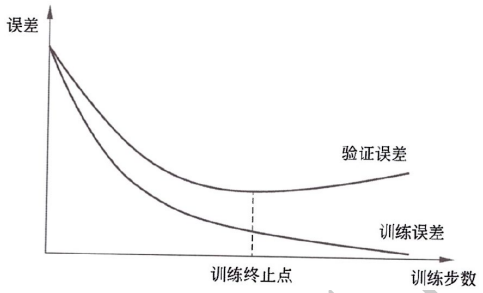


层归一化的网络结构

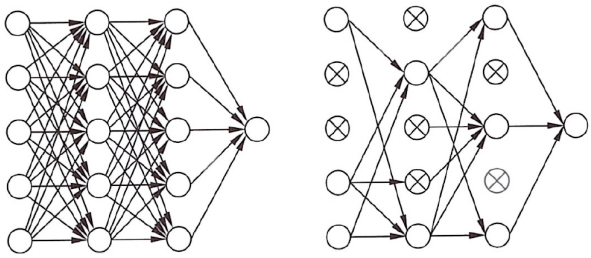
12.1.3 过拟合问题

深度学习解决过拟合的方法为**正则化**，有**权重衰减**（L1/L2 正则化）、**早停法**、**暂退法**等。而深度学习又有特殊之处——常常不做正则化也不产生过拟合，这往往发生于大规模训练数据 + 过度参数化神经网络 + 随机梯度下降训练；目前普遍解释是认为随机梯度下降起到隐式正则化的作用，能保证学到的模型不产生过拟合。

早停法：在学习中使用验证集进行评估，判断训练的终止点，进行模型选择，是**隐式的正则化方法**。即可将数据分为训练集、验证集、测试集（0.5，0.25，0.25 的比例），持续训练模型的过程中用验证集进行评估，得到验证误差，选择验证误差最小的点停止训练。这里的“模型复杂度”不再是模型本身架构的复杂度，而是“**模型训练程度**”。



暂退法：训练过程中的每一步**随机选取一些神经元**，让它们**不参与（退出）训练**，学习结束后，对权重进行调整，然后将整体网络用于预测。属于经验性的方法，有效但无严格的理论证明，可以认为是应用于深度学习的一种 Bagging 方法。

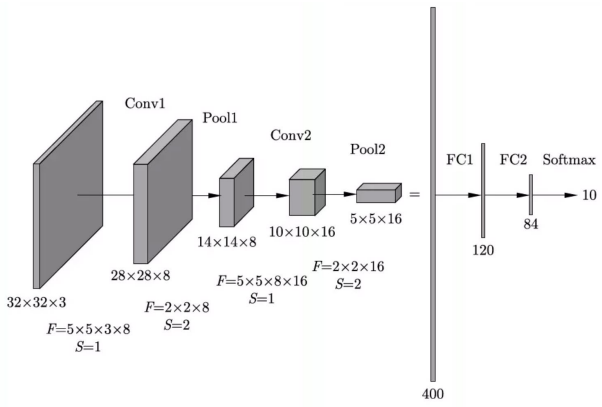


- 设输入层和隐层的每一层都有一个保留概录 p：通常输入层 p= 0.8 , 隐层 p = 0.5；
- 训练的每一步（一次前向后向传播），对每一个样本，在每一层随机判断，选取保留的神经元；
- 所有保留的神经元构成了一个退化的神经网络，也是整体网络的一个子网络；
- 随机梯度下降法更新子网络的权重；
- 训练完成后，对隐层权重 w 乘以保留概率 p 作为最终的权重。

12.2 卷积神经网络的基本操作

卷积神经网络（CNN）是一种专门为处理具有网格结构数据（如图像、音频）而设计的深度学习模型，通过卷积层中的卷积核与输入数据进行卷积操作，自动提取数据中的特征。

特征：层次化网络结构；前一层的输出是后一层的输入；前面几层每一层进行 **卷积运算**（实现特征检测）或**汇聚运算**（实现特征选择）；最后几层是全连接的前馈神经网络，进行分类或回归。



来源：典型的图像数据是灰度图像，可以用**实数矩阵**表示，在使用**前馈神经网络**时是将图像的矩阵数据展开成一个很长的**向量**作为输入，当输入向量维度很高时，用全连接则参数量太大，难以很好地学习模型，并且图像在不同位置上有相似的**局部特征**，前馈神经网络对不同位置的局部特征是**分开表示和学习**的，产生冗余，会降低表示和学习的效率。**卷积神经网络**则是从生物视觉系统得到启发，在前馈神经网络中导入 **卷积运算**解决问题。

12.2.1 卷积：特征检测

数学中的卷积（了解）——定义在两个函数上的运算，表示用其中一个函数对另一个函数的形状进行的调整。

* **一维卷积**：设 f 与 g 是两个可积的实值函数，$\circledast$ 表示卷积运算，则 f 与 g 的卷积为

$$h(t) = (f \circledast g)(t) = \int_{-\infty}^{+\infty} f(\tau) g(t-\tau)\,\mathrm{d}\tau$$

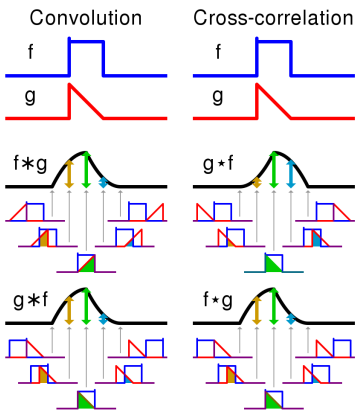
卷积满足交换律 $(f \circledast g)(t) = (g \circledast f)(t)$

卷积神经网络中的卷积——实际是数学中的**互相关**，和严格的数学卷积定义不同。

* **一维互相关**：设 f 和 g 是两个可积的实值函数，则他们的互相关函数为（* 表示互相关运算）：

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau) g(t+\tau)\,\mathrm{d}\tau$$

互相关不满足交换律，互相关拓展至二维和离散情况就是卷积神经网络中的卷积



如上图，左图展示了卷积的交换律，数学卷积公式中的“减”表示翻转后再相乘，而右图展示了互相关交换后并不相等，因为公式中的“加”表示平移直接相乘，因此交换后结果是对称的。

二维卷积（convolution）：给定一个 $I \times J$ 输入矩阵 $\boldsymbol{X} = \left[x_{ij}\right]_{I\times J}$，一个 $M \times N$ 核矩阵 $\boldsymbol{W} = \left[w_{mn}\right]_{M\times N}$，满足 $M \ll I, N \ll J$，让核矩阵在输入矩阵上从左到右再从上到下按顺序滑动，在滑动的每一个位置，核矩阵与输入矩阵的一个子矩阵重叠。求核矩阵与每一个子矩阵的内积，产生一个 $K \times L$ 输出矩阵 $\boldsymbol{Y} = \left[y_{kl}\right]_{K\times L}$，称此运算为**卷积**或**二维卷积**。写作

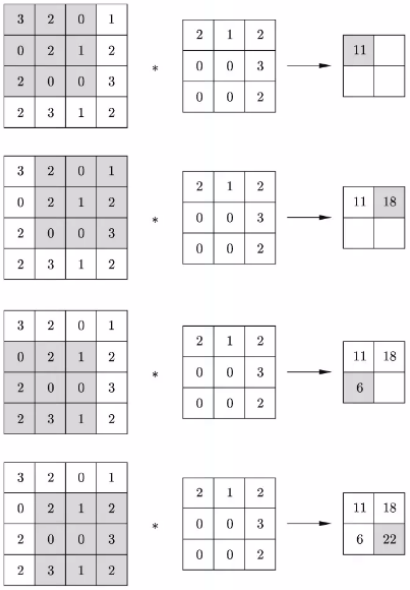
$$\boldsymbol{Y} = \boldsymbol{X} * \boldsymbol{W}$$

其中，$\boldsymbol{Y} = \left[y_{kl}\right]_{K\times L}$，

$$y_{kl} = \sum_{m=1}^{M}\sum_{n=1}^{N} w_{m,n}x_{k+m-1,l+n-1}$$

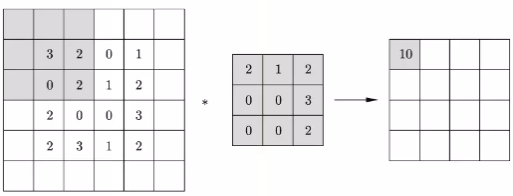
其中，$k = 1, 2, \ldots, K,\ l = 1, 2, \ldots, L,\ K = I - M + 1,\ L = J - N + 1$。

如下图示例可以很好地理解二维卷积的计算过程：即核矩阵 W 去依次平移覆盖输入矩阵 X，两者的相同位置元素相乘，最后求和得到输出矩阵 Y 的元素。

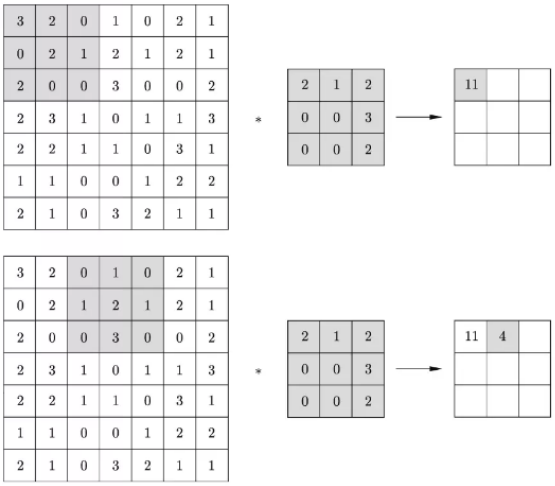


卷积运算的扩展可以通过增加填充和步幅实现。

- **（零）填充**：在输入矩阵的周边添加元素为 0 的行和列，使卷积核能更充分地作用于**输入矩阵边缘**的元素。如下图在左侧的输入矩阵增加了多个元素 0，使得原本输入矩阵边缘的元素能多次被覆盖。



- **步幅**：卷积运算中卷积核每次向右或者向下移动的列数或行数。先前都是每次平移一行/一列，如下图当步幅为 2 时则是平移 2 行/列。



卷积运算依赖于卷积运算的超参数——**卷积核的大小**、**填充**的大小、**步幅**。假设输入矩阵的大小是 $I \times J$，卷积核的大小是 $M \times N$，两个方向填充的大小是 P 和 Q，步幅的大小是 S，则卷积的输出矩阵的大小 $K \times L$ 满足

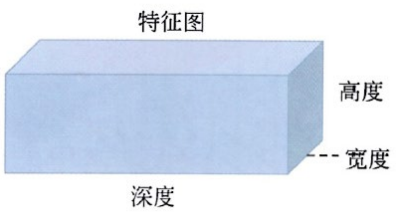
$$K = \left\lfloor \frac{I + 2P - M}{S} + 1 \right\rfloor, \quad L = \left\lfloor \frac{J + 2Q - N}{S} + 1 \right\rfloor$$

这里 $\lfloor a \rfloor$ 表示不超过 a 的最大整数。填充 P 和 Q 的最大值分别是 $M-1$ 和 $N-1$，这时的填充称为**全填充**。

卷积运算的**物理含义**——输入矩阵表示灰度图像，核矩阵表示特征（如物体的边缘），核矩阵的每个元素代表特征在一个像素点上的灰度（权重）。卷积核平移的过程则是在图像的每一个位置对一个特定的特征进行检测并输出一个检测值，特征越一致，检测值也越大。卷积核的权重是通过学习获得的。卷积的输入和输出称为**特征图**。

三维卷积：输入和输出是由**张量**表示的特征图。

- **张量**：多个大小相同的矩阵组成，矩阵的行数和列数是张量的高度和宽度，矩阵的个数是张量的深度。
- 矩阵表示的特征图可以看作是一张特征图，张量表示的特征图可以看作是多张特征图。
- **彩色图像数据**就是一种张量特征图，由红、绿、蓝三个通道的数据组成，每一个通道的数据由一个矩阵表示，三个矩阵排列起来构成一个张量。



如下示例展示了三维卷积的处理过程：

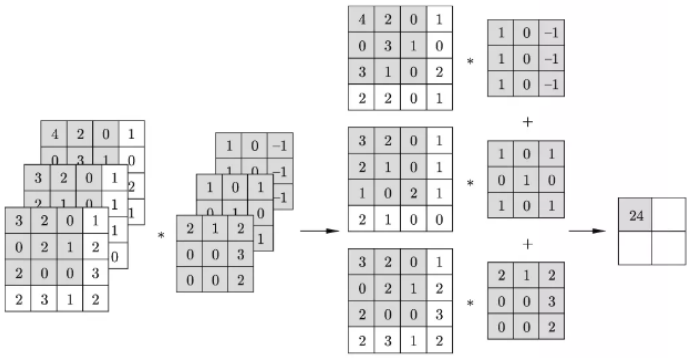
输入张量由三个通道的矩阵组成 $\boldsymbol{X}=(\boldsymbol{X}_{\mathrm{R}},\boldsymbol{X}_{\mathrm{G}},\boldsymbol{X}_{\mathrm{B}})$，

$$\boldsymbol{X}_{\mathrm{R}}=\begin{bmatrix}3&2&0&1\\0&2&1&2\\2&0&0&3\\2&3&1&2\end{bmatrix},\quad \boldsymbol{X}_{\mathrm{G}}=\begin{bmatrix}3&2&0&1\\2&1&0&1\\1&0&2&1\\2&1&0&0\end{bmatrix},\quad \boldsymbol{X}_{\mathrm{B}}=\begin{bmatrix}4&2&0&1\\0&3&1&0\\3&1&0&2\\2&2&0&1\end{bmatrix}$$

卷积核张量由三个矩阵组成 $\boldsymbol{W}=(\boldsymbol{W}_{\mathrm{R}},\boldsymbol{W}_{\mathrm{G}},\boldsymbol{W}_{\mathrm{B}})$，

$$\boldsymbol{W}_{\mathrm{R}}=\begin{bmatrix}2&1&2\\0&0&3\\0&0&2\end{bmatrix},\quad \boldsymbol{W}_{\mathrm{G}}=\begin{bmatrix}1&0&1\\0&1&0\\1&0&1\end{bmatrix},\quad \boldsymbol{W}_{\mathrm{B}}=\begin{bmatrix}1&0&-1\\1&0&-1\\1&0&-1\end{bmatrix}$$

求在其上的三维卷积 $\boldsymbol{Y}$。



每个输入矩阵与对应的卷积核进行一维卷积后，再整体相加得到输出特征图的元素，在一个卷积核 $\boldsymbol{W}=(\boldsymbol{W}_{\mathrm{R}},\boldsymbol{W}_{\mathrm{G}},\boldsymbol{W}_{\mathrm{B}})$ 处理下得到的是一个输出矩阵特征图，多个卷积核并行重复次操作得到张量特征图。

12.2.2 汇聚：特征选择

汇聚（pooling）：给定一个 $I\times J$ 输入矩阵 $\boldsymbol{X}=\left[x_{ij}\right]_{I\times J}$，一个虚设的 $M\times N$ 核矩阵，$M\ll I, N\ll J$。让核矩阵在输入矩阵上从左到右再从上到下滑动，将输入矩阵划分成若干大小为 $M\times N$ 的子矩阵，这些子矩阵相互不重叠且完全覆盖整个输入矩阵。对每一个子矩阵求最大值或平均值，产生一个 $K\times L$ 输出矩阵 $\boldsymbol{Y}=\left[y_{kl}\right]_{K\times L}$，称此运算为汇聚或二维汇聚。对子矩阵取最大值的称为**最大汇聚**，取平均值的称为**平均汇聚**。即有

$$y_{kl}=\max_{\substack{m\in\{1,2,\ldots,M\}\\ n\in\{1,2,\ldots,N\}}} x_{k+m-1,l+n-1}\quad \text{或}$$

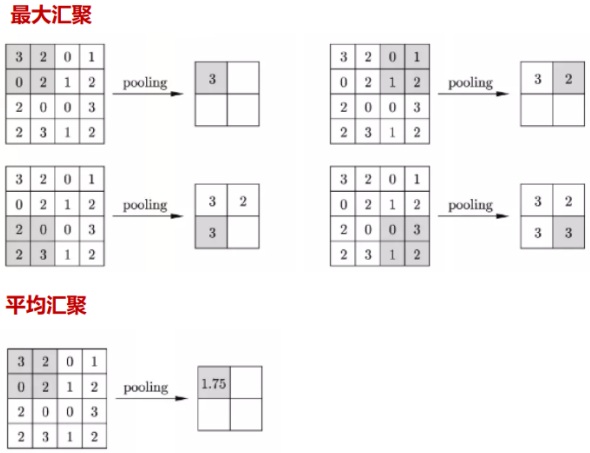
$$y_{kl} = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} x_{k+m-1,l+n-1}$$

其中, $k = 1, 2, \ldots, K$, $l = 1, 2, \ldots, L$, K 和 L 满足

$$K = \frac{I}{M}, \quad L = \frac{J}{N}$$

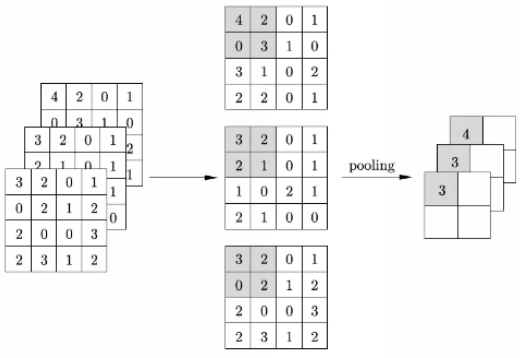
这里假设 I 和 J 分别可以被 M 和 N 整除。

如矩阵 $\boldsymbol{X} = \begin{bmatrix} 3 & 2 & 0 & 1 \\ 0 & 2 & 1 & 2 \\ 2 & 0 & 0 & 3 \\ 2 & 3 & 1 & 2 \end{bmatrix}$，选取核函数大小为 2*2，步幅为 2，下图展示了上述汇聚的过程。



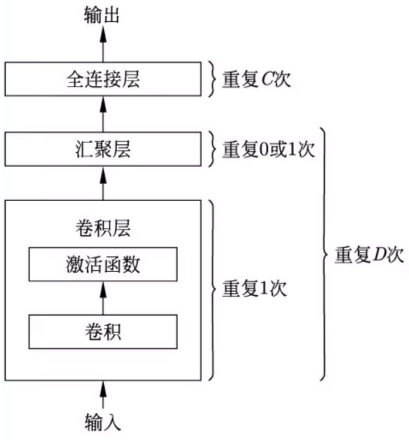
汇聚的**物理含义**——汇聚的输入是矩阵，每个元素表示特征的检测值，即卷积的输出；汇聚运算是将汇聚核在输入矩阵上进行滑动，从汇聚核覆盖的特征检测值中选择一个最大值或平均值，这样可以有效地进行**特征抽取**；输出一个缩小的矩阵，也就是进行了下采样。

三维汇聚：汇聚对输入张量的各个矩阵分别进行汇聚计算，再将结果排列起来，产生输出张量。汇聚的输出特征图高度和宽度比输入特征图更小，但是两者深度相同，如下图详细展示了此过程。



12.3 卷积神经网络的模型

卷积神经网络是包含**卷积**与**汇聚**运算的一种特殊的前馈神经网络。其输入是**张量**，输出是标量，表示分类或回归的预测值。中间层每层的输入和输出都是张量（包括矩阵）表示的特征图，中间层可分为**卷积层、汇聚层、全连接层**。



卷积层：进行**基于卷积的仿射变换**和**基于激活函数的非线性变换**。假设第 l 层是卷积层，则第 l 层的计算如下：

$$\boldsymbol{Z}^{(l)} = \boldsymbol{W}^{(l)} * \boldsymbol{X}^{(l-1)} + \boldsymbol{b}^{(l)}$$

$$\boldsymbol{X}^{(l)} = a\big(\boldsymbol{Z}^{(l)}\big)$$

输入 $\boldsymbol{X}^{(l-1)}$：$I \times J \times K$ 张量，　输出 $\boldsymbol{X}^{(l)}$：$I' \times J' \times K'$ 张量

卷积核 $\boldsymbol{W}^{(l)}$：$M \times N \times K \times K'$ 张量，　偏置 $\boldsymbol{b}^{(l)}$：$I' \times J' \times K'$ 张量

净输入 $\boldsymbol{Z}^{(l)}$：$I' \times J' \times K'$ 张量，　$a(\cdot)$ 是激活函数。

矩阵表示：将 $I \times J \times K$ 张量展开成 K 个 $I \times J$ 矩阵

$$\boldsymbol{Z}_{k'}^{(l)} = \sum_{k} \boldsymbol{W}_{k,k'}^{(l)} * \boldsymbol{X}_{k}^{(l-1)} + \boldsymbol{b}_{k'}^{(l)}$$

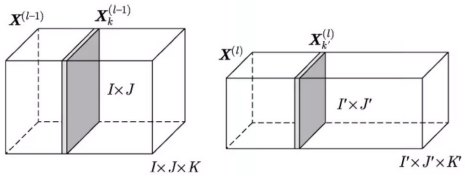
$$\boldsymbol{X}_{k'}^{(l)} = a\big(\boldsymbol{Z}_{k'}^{(l)}\big)$$

$\boldsymbol{X}_{k}^{(l-1)}$：输入的第 k 个 $I \times J$ 矩阵，　$\boldsymbol{X}_{k'}^{(l)}$：输出的第 k' 个 $I' \times J'$ 矩阵

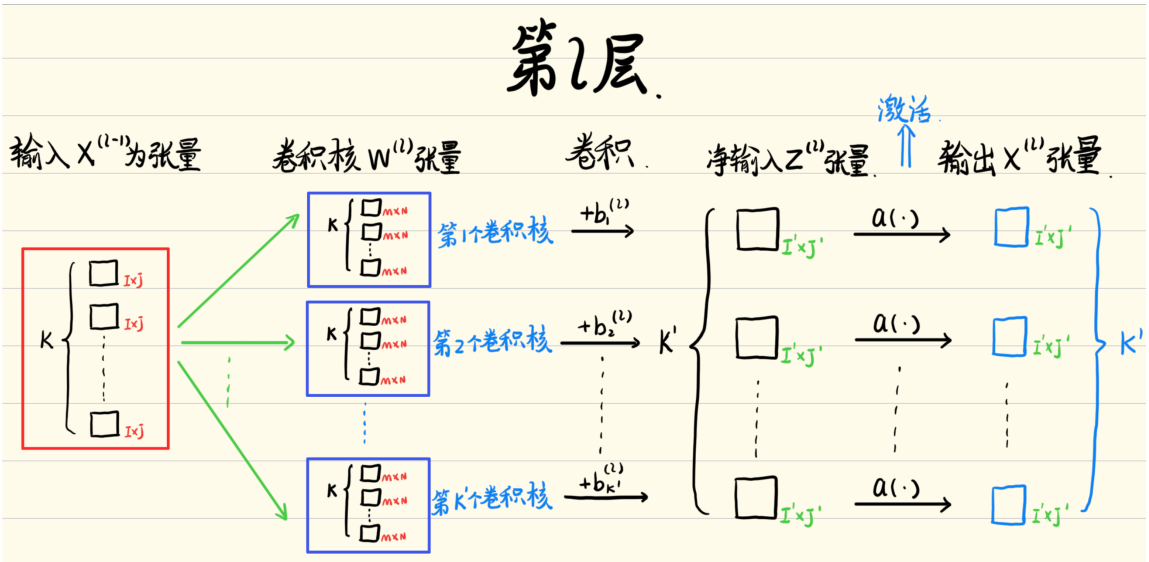
$\boldsymbol{W}_{k,k'}^{(l)}$：卷积核的第 $k \times k'$ 个 $M \times N$ 矩阵，　$\boldsymbol{b}_{k'}^{(l)}$：偏置的第 k' 个 $I' \times J'$ 矩阵

$\boldsymbol{Z}_{k'}^{(l)}$：净输入的第 k' 个 $I' \times J'$ 矩阵。

如下图展示了卷积层操作前后张量的变化。



或许如下图展开能帮助我们更好地理解这个公式计算的全过程：



汇聚层：假设第 l 层是汇聚层，则：

$$\boldsymbol{X}^{(l)} = \text{pooling}\left(\boldsymbol{X}^{(l-1)}\right)$$

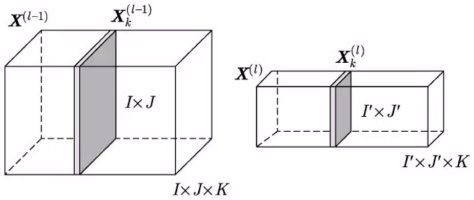
输入 $\boldsymbol{X}^{(l-1)}$：$I \times J \times K$ 张量，输出 $\boldsymbol{X}^{(l)}$：$I' \times J' \times K$ 张量，$\text{pooling}(\cdot)$ 是汇聚运算。

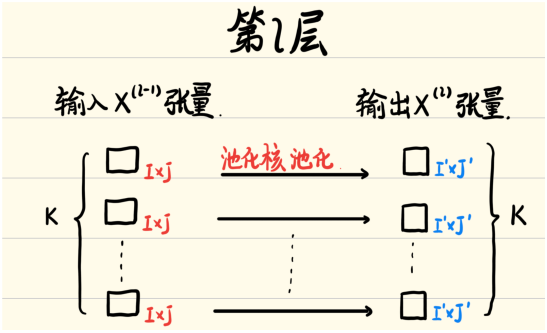
矩阵表示：将 $I \times J \times K$ 张量展开成 K 个 $I \times J$ 矩阵，将 $I' \times J' \times K'$ 张量展开成 K' 个 $I' \times J'$ 矩阵：

$$\boldsymbol{X}_k^{(l)} = \text{pooling}\left(\boldsymbol{X}_k^{(l-1)}\right)$$

$\boldsymbol{X}_k^{(l-1)}$：输入的第 k 个 $I \times J$ 矩阵，$\boldsymbol{X}_k^{(l)}$：输出的第 k 个 $I' \times J'$ 矩阵。

同样下图展示了汇聚层的张量变化过程。相比卷积，池化的全过程也更加简单。





全连接层：全连接的第 l 层是前馈神经网络的一层，进行仿射变换和非线性变换：

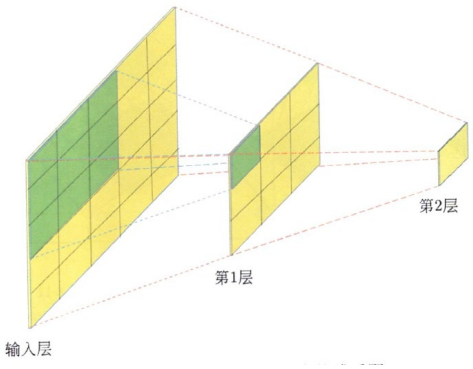
$$\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{x}^{(l-1)} + \boldsymbol{b}^{(l)}$$
$$\boldsymbol{x}^{(l)} = a\left(\boldsymbol{z}^{(l)}\right)$$

$\boldsymbol{x}^{(l-1)}$：N 维输入向量，由张量展开得到，$\boldsymbol{x}^{(l)}$：是 M 维输出向量
$\boldsymbol{W}^{(l)}$：$M \times N$ 权重矩阵，$\boldsymbol{b}^{(l)}$：M 维偏置向量
$\boldsymbol{z}^{(l)}$：M 维净输入向量，$a(\cdot)$ 是激活函数

卷积神经网络中的**参数**包括**卷积核的权重和偏置**与**全连接的权重和偏置**，两者都通过学习获得。CNN 也可以只有卷积层和全连接层，而**没有汇聚层**（步幅大于 1 的卷积运算也可以起到下采样作用，以代替汇聚运算）。为了达到更好的预测效果，设计上使用更小的卷积核和更深的结构，前端使用少量的卷积核，后端使用大量的卷积核。

卷积神经网络的**性质**

- **表示效率高**：CNN 的表示和学习效率比前馈神经网络高。卷积代表的是稀疏连接，比全连接的数目大幅减少；卷积核在前一层的各个位置上滑动计算，在所有位置上具有相同的参数，大幅减少了参数的数量；每一层内的卷积运算可以并行处理，加快学习和推理的速度。
- **平移不变性**：**不变性**——设 $f(x)$ 是以 x 为输入的函数，$\tau(x)$ 是对 x 的变换，如平移、旋转变换、缩放变换。若满足关系 $f(x) = f\left(\tau(x)\right)$，则称函数 $f(\cdot)$ 对变换 $\tau(\cdot)$ 具有不变性。CNN 具有平移不变性，但不能严格保证。CNN 不具有旋转不变性、缩放不变性。
- **层级化感受野**：CNN 利用卷积实现了图像处理需要的特征的表示。前端神经元表示**局部特征**（只需要少数卷积核），后端神经元表示的是**全局的特征**（需要较多卷积核）。**感受野**——神经元涵盖的输入矩阵的部分。如下图，第一层的每个神经元是输入层 3*3 卷积得到的，因此每个神经元的感受野就是输入层的 3*3 区域；第二层池化核为 3*3 区域，得到最大的值作为第二层的神经元，则这个神经元的感受野是整个输入层。



13 循环神经网络

在先前所学习的普通的神经网络中，信息的传递是单向的，这让网络变得更加容易学习。但是在一些现实任务中，网络的输出不仅和当前时刻的输入相关，也和过去的一段时间的输出相关（如输入“What time is it”，需要按照顺序读取单词并且每次读取不是“孤立地”，而是将所有词联系起来才能理解这句话）。同时，前馈神经网络要求输入和输出的维数都是固定的，不能任意改变，难以处理长度不固定的时序数据。由此处理这种问题需要循环神经网络。

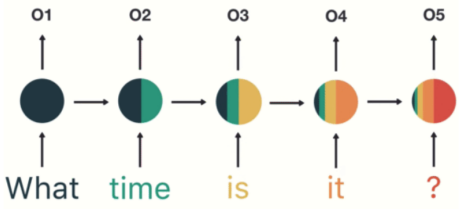
13.1 简单循环神经网络

循环神经网络（RNN）是一种具有短期记忆能力，能够处理**序列数据**（如文本、语音、时间序列）的神经网络模型。**循环**指序列数据的每一个位置上具有相同的结构。

思路：

- 在序列数据的每一个位置上**重复使用相同的前馈神经网络**；
- 用前馈神经网络隐层的输出表示当前位置具有**马尔科夫性质**（下一个状态只和当前状态有关，和更早的状态无关）的“隐状态”；
- 通过**隐状态**将相邻位置的神经网络**连接**起来；
- 总体表示和学习序列数据中的**局部和全局特征**。

如下例子的循环神经网络结构，每个位置上的隐状态都会“记得”先前的信息：



定义：神经网络在每一个位置上重复使用同一个神经网络结构。在第 t 个位置上 $(t=1,2,\cdots,T)$，神经网络的隐层以 $\boldsymbol{x}_t$ 和 $\boldsymbol{h}_{t-1}$ 为输入，以 $\boldsymbol{h}_t$ 为输出，其间有以下关系成立：

$$\boldsymbol{h}_t = \tanh(\boldsymbol{U}\cdot\boldsymbol{h}_{t-1} + \boldsymbol{W}\cdot\boldsymbol{x}_t + \boldsymbol{b})$$

其中，$\boldsymbol{x}_t$ 表示第 t 个位置上的输入，是一个实数向量；$\boldsymbol{h}_{t-1}$ 表示第 $t-1$ 个位置的状态，也是一个实数向量；$\boldsymbol{h}_t$ 表示第 t 个位置的状态，也是一个实数向量；tanh 为激活函数，把结果压缩到-1 到 1 之间；$\boldsymbol{U},\boldsymbol{W}$ 是权重矩阵；$\boldsymbol{b}$ 是偏置向量。神经网络的输出层以 $\boldsymbol{h}_t$ 为输入，$\boldsymbol{p}_t$ 为输出，有以下关系成立：

$$\boldsymbol{p}_t = \text{softmax}(\boldsymbol{V}\cdot\boldsymbol{h}_t + \boldsymbol{c})$$

其中，softmax 表示把结果转成概率分布的函数；$\boldsymbol{p}_t$ 表示第 t 个位置上的输出，是一个概率向量；$\boldsymbol{V}$

是权重矩阵; $\boldsymbol{c}$ 是偏置向量。神经网络输出序列数据 $\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_T$, 每一项是一个概率向量。

如下展示了简单循环神经网络的结构图:

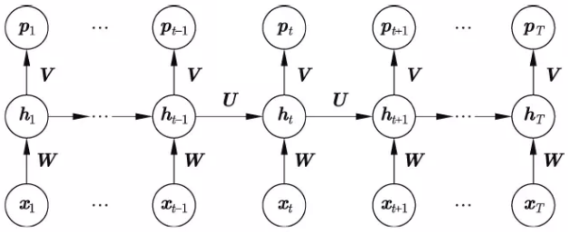


图 25.1 简单循环神经网络架构（展开形式）
偏置向量被省去

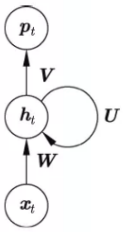


图 25.2 简单循环神经网络架构（折叠形式）
偏置向量被省去

$$\boldsymbol{r}_t = \boldsymbol{U} \cdot \boldsymbol{h}_{t-1} + \boldsymbol{W} \cdot \boldsymbol{x}_t + \boldsymbol{b}$$

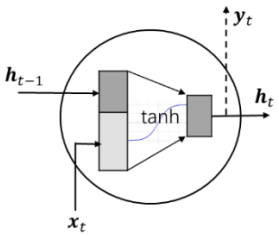
$$\boldsymbol{h}_t = \tanh(\boldsymbol{r}_t)$$

$$\boldsymbol{z}_t = \boldsymbol{V} \cdot \boldsymbol{h}_t + \boldsymbol{c}$$

$$\boldsymbol{p}_t = \text{softmax}(\boldsymbol{z}_t)$$

$\boldsymbol{r}_t$ 是隐层的净输入向量，$\boldsymbol{h}_t$ 是隐状态，$\boldsymbol{z}_t$ 是输出层的净输入向量；隐层的激活函数通常是双曲正切函数，也可以是其他激活函数；输出层的激活函数通常是软最大化函数

单元：循环神经网络的每一个位置上, 有以**当前位置的输入和之前位置的状态**为 输入 、以**当前位置的状态**为 输出 的函数, 称为**单元**，是核心处理模块。如下图为基本 RNN 单元



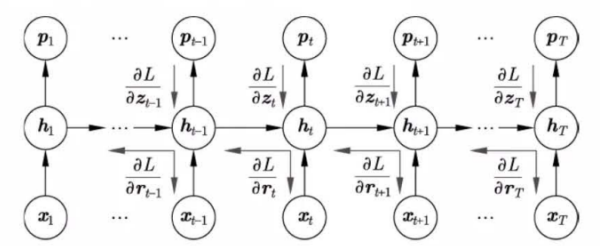
模型**特点**:

- 马尔科夫性隐状态 $\boldsymbol{h}_t$：当前位置的状态 $\boldsymbol{h}_t$ 由当前位置的输入 $\boldsymbol{x}_t$ 和之前位置的状态 $\boldsymbol{h}_{t-1}$ 决定；每

一个位置的状态表示的是到这个位置为止的序列数据的**局部特征**及**全局特征**，也称作短距离依存关系和长距离依存关系。

- 属于**自回归模型**：序列数据的每一个位置上的预测只使用之前位置的信息；计算需要在序列数据上依次进行，可以处理任意长度的序列数据但是不能进行并行化处理以提高计算效率。
- RNN 具有强大的表示能力，是动态系统的通用模型。动态系统的核心是状态之间按时间顺序的依存关系。

* 了解——RNN 随时间的**反向传播**算法



损失函数 L 对各参数的偏导数：

$$\frac{\partial L}{\partial \boldsymbol{b}} = \sum_{t=1}^{T} \frac{\partial \boldsymbol{r}_t}{\partial \boldsymbol{b}} \frac{\partial L}{\partial \boldsymbol{r}_t} = \sum_{t=1}^{T} \frac{\partial L}{\partial \boldsymbol{r}_t}$$

$$\frac{\partial L}{\partial \boldsymbol{U}} = \sum_{t=1}^{T} \frac{\partial \boldsymbol{r}_t}{\partial \boldsymbol{U}} \frac{\partial L}{\partial \boldsymbol{r}_t} = \sum_{t=1}^{T} \frac{\partial L}{\partial \boldsymbol{r}_t} \cdot \boldsymbol{h}_{t-1}^T$$

$$\frac{\partial L}{\partial \boldsymbol{W}} = \sum_{t=1}^{T} \frac{\partial \boldsymbol{r}_t}{\partial \boldsymbol{W}} \frac{\partial L}{\partial \boldsymbol{r}_t} = \sum_{t=1}^{T} \frac{\partial L}{\partial \boldsymbol{r}_t} \cdot \boldsymbol{x}_t^T$$

$$\frac{\partial L}{\partial \boldsymbol{c}} = \sum_{t=1}^{T} \frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{c}} \frac{\partial L}{\partial \boldsymbol{z}_t} = \sum_{t=1}^{T} \frac{\partial L}{\partial \boldsymbol{z}_t}$$

$$\frac{\partial L}{\partial \boldsymbol{V}} = \sum_{t=1}^{T} \frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{V}} \frac{\partial L}{\partial \boldsymbol{z}_t} = \sum_{t=1}^{T} \frac{\partial L}{\partial \boldsymbol{z}_t} \cdot \boldsymbol{h}_t^T$$

$$\frac{\partial L}{\partial \boldsymbol{z}_t} = \frac{\partial L}{\partial L_t} \frac{\partial L_t}{\partial \boldsymbol{z}_t} = \boldsymbol{y}_t - \boldsymbol{p}_t,\ t = 1, 2, \cdots, T$$

第 t 个位置的偏导数 $\dfrac{\partial L}{\partial \boldsymbol{r}_t}$ 需要通过 $\dfrac{\partial L}{\partial \boldsymbol{r}_{t+1}}$ 和 $\dfrac{\partial L}{\partial \boldsymbol{z}_t}$ 计算：

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{r}_t} &= \frac{\partial \boldsymbol{r}_{t+1}}{\partial \boldsymbol{r}_t} \frac{\partial L}{\partial \boldsymbol{r}_{t+1}} + \frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{r}_t} \frac{\partial L}{\partial \boldsymbol{z}_t} \\ &= \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{r}_t} \frac{\partial \boldsymbol{r}_{t+1}}{\partial \boldsymbol{h}_t} \frac{\partial L}{\partial \boldsymbol{r}_{t+1}} + \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{r}_t} \frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{h}_t} \frac{\partial L}{\partial \boldsymbol{z}_t} \\ &= \operatorname{diag}(1 - \tanh^2 \boldsymbol{r}_t) \cdot \boldsymbol{U}^\top \cdot \frac{\partial L}{\partial \boldsymbol{r}_{t+1}} \\ &\quad + \operatorname{diag}(1 - \tanh^2 \boldsymbol{r}_t) \cdot \boldsymbol{V}^\top \cdot \frac{\partial L}{\partial \boldsymbol{z}_t}\end{aligned}$$

13.2 常用循环神经网络

简单循环神经网络的反向传播的计算依赖矩阵 $A_t = \operatorname{diag}(1 - \tanh^2 \boldsymbol{r}_t) \cdot \boldsymbol{U}^\top$ 的连乘，有可能使得到的矩阵的一些元素趋近 0 或无穷大，存在梯度消失与梯度放大的问题。为了避免这个问题，可以使用 **LSTM** 和 **GRU**。

1）长短期记忆网络（LSTM）

简单循环神经网络中 $\boldsymbol{h}_t = \tanh(\boldsymbol{U} \cdot \boldsymbol{h}_{t-1} + \boldsymbol{W} \cdot \boldsymbol{x}_t + \boldsymbol{b})$，隐状态虽然能同时表示序列数据中的**短距离**和**长距离**依存关系，并对短距离依存关系可以有效地表示和学习，但是**对长距离依存关系的处理能力有限**，因为长距离依存关系在模型中会被逐渐“遗忘”。

LSTM 解决问题的**思路**：记录并使用之前所有位置的状态，更好描述短距离和长距离依存关系。

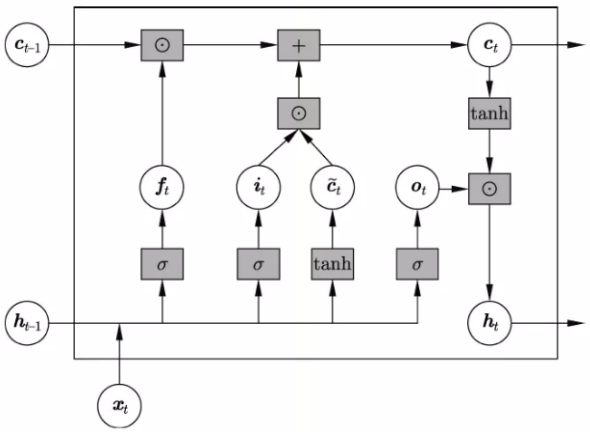
- **记忆元**：用于记录之前位置的状态信息；
- **门控**：用**门函数**来控制状态信息的使用，分遗忘门、输入门、输出门；
- **长（的）短期记忆**：状态的表示包括对远距离的状态的“记忆”。

定义：在循环网络的每一个位置上有状态和记忆元，以及输入门、遗忘门、输出门，构成一个单元。第 t 个位置上 $(t=1,2,\cdots,T)$ 的单元是以当前位置的输入 $\boldsymbol{x}_t$、之前位置的记忆元 $\boldsymbol{c}_{t-1}$、之前位置的状态 $\boldsymbol{h}_{t-1}$ 为输入，以当前位置的状态 $\boldsymbol{h}_t$ 和当前位置的记忆元 $\boldsymbol{c}_t$ 为输出的函数，由以下方式计算。

$$
\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{U}_i \cdot \boldsymbol{h}_{t-1} + \boldsymbol{W}_i \cdot \boldsymbol{x}_t + \boldsymbol{b}_i) \\
\boldsymbol{f}_t &= \sigma(\boldsymbol{U}_f \cdot \boldsymbol{h}_{t-1} + \boldsymbol{W}_f \cdot \boldsymbol{x}_t + \boldsymbol{b}_f) \\
\boldsymbol{o}_t &= \sigma(\boldsymbol{U}_o \cdot \boldsymbol{h}_{t-1} + \boldsymbol{W}_o \cdot \boldsymbol{x}_t + \boldsymbol{b}_o) \\
\tilde{\boldsymbol{c}}_t &= \tanh(\boldsymbol{U}_c \cdot \boldsymbol{h}_{t-1} + \boldsymbol{W}_c \cdot \boldsymbol{x}_t + \boldsymbol{b}_c) \\
\boldsymbol{c}_t &= \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t)
\end{aligned}
$$

这里 σ 为 sigmoid 激活函数，$\boldsymbol{i}_t$ 是输入门（决定从之前位置传入哪些信息），$\boldsymbol{f}_t$ 是遗忘门（决定忘记之前位置的哪些信息），$\boldsymbol{o}_t$ 是输出门（决定向下一个位置传出哪些信息），$\tilde{\boldsymbol{c}}_t$ 是中间结果。状态 $\boldsymbol{h}_t$、记忆元 $\boldsymbol{c}_t$、输入门 $\boldsymbol{i}_t$、遗忘门 $\boldsymbol{f}_t$、输出门 $\boldsymbol{o}_t$ 都是向量，其维度相同。

如下为 LSTM 单元示意图：



LSTM 模型**特点**：

- 记忆元、输入门和遗忘门起着重要作用，模型能更好地表示和学习**长距离**依存关系。

  当输入门 $i_t=1$，遗忘门 $f_t=0$ 时，当前位置的记忆元 c_t 只依赖输入 $\boldsymbol{x}_t$ 和之前位置的状态 $\boldsymbol{h}_{t-1}$，LSTM 是基本的 RNN 近似；

当输入门 $i_t = 0$，遗忘门 $f_t = 1$ 时，当前位置的记忆元 c_t 只依赖于之前位置的记忆元 c_{t-1}，LSTM 将之前位置的神经元复制到当前位置。

当前位置的记忆元 c_t 可展开为：

$$\boldsymbol{c}_t = \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} = \sum_{i=1}^{t} \left(\prod_{j=i+1}^{t} \boldsymbol{f}_j \odot \boldsymbol{i}_i \right) \odot \tilde{\boldsymbol{c}}_i = \sum_{i=1}^{t} \boldsymbol{w}_i^t \odot \tilde{\boldsymbol{c}}_i$$

推导：

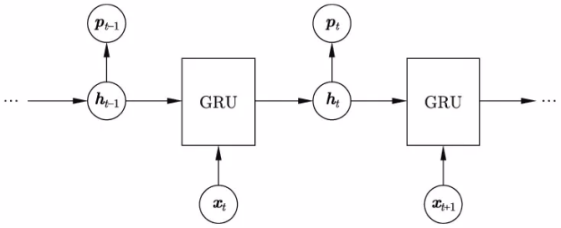
$$\begin{aligned}
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \\
&= \boldsymbol{f}_t \odot (\boldsymbol{f}_{t-1} \odot \boldsymbol{c}_{t-2} + \boldsymbol{i}_{t-1} \odot \tilde{\boldsymbol{c}}_{t-1}) + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \\
&= \boldsymbol{f}_t \odot \boldsymbol{f}_{t-1} \odot \boldsymbol{c}_{t-2} + \boldsymbol{f}_t \odot \boldsymbol{i}_{t-1} \odot \tilde{\boldsymbol{c}}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \\
&= \boldsymbol{f}_t \odot \boldsymbol{f}_{t-1} \odot (\boldsymbol{f}_{t-2} \odot \boldsymbol{c}_{t-3} + \boldsymbol{i}_{t-2} \odot \tilde{\boldsymbol{c}}_{t-2}) + \boldsymbol{f}_t \odot \boldsymbol{i}_{t-1} \odot \tilde{\boldsymbol{c}}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \\
&= \boldsymbol{f}_t \odot \boldsymbol{f}_{t-1} \odot \boldsymbol{f}_{t-2} \odot \boldsymbol{c}_{t-3} + \boldsymbol{f}_t \odot \boldsymbol{f}_{t-1} \odot \boldsymbol{i}_{t-2} \odot \tilde{\boldsymbol{c}}_{t-2} + \boldsymbol{f}_t \odot \boldsymbol{i}_{t-1} \odot \tilde{\boldsymbol{c}}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \\
&\vdots \\
&= \left(\prod_{j=1}^{t} \boldsymbol{f}_j \right) \odot \boldsymbol{c}_0 + \sum_{i=1}^{t} \left(\prod_{j=i+1}^{t} \boldsymbol{f}_j \odot \boldsymbol{i}_i \right) \odot \tilde{\boldsymbol{c}}_i \\
&= \sum_{i=1}^{t} \left(\prod_{j=i+1}^{t} \boldsymbol{f}_j \odot \boldsymbol{i}_i \right) \odot \tilde{\boldsymbol{c}}_i
\end{aligned}$$

令 $\boldsymbol{w}_i^t = \prod_{j=i+1}^{t} \boldsymbol{f}_j \odot \boldsymbol{i}_i$，得到：$\boldsymbol{c}_t = \sum_{i=1}^{t} \boldsymbol{w}_i^t \odot \tilde{\boldsymbol{c}}_i$

- **避免梯度消失与爆炸**：位置之间的梯度传播不是通过矩阵的连乘而是通过**矩阵连乘的线性组合**，所以可以避免梯度消失和梯度爆炸。

2）门控循环网络（GRU）

与 LSTM 相比，GRU 只有 2 个门控——**更新门**（相当于 LSTM 合并的输入门和遗忘门）和**重置门**。GRU 没有了记忆元，而是直接用隐状态 $\boldsymbol{h}_t$ 来承载“长期记忆”，把这个筛选记忆的功能交给了重置门 $\boldsymbol{r}_t$（控制上一时刻的旧记忆 $\boldsymbol{h}_{t-1}$ 对当前新候选状态 $\tilde{\boldsymbol{h}}_t$ 的影响程度）。



在循环网络的每一个位置上有状态及重置门、更新门，构成一个单元。第 t 个位置上 $(t=1,2,\cdots,T)$ 的单元是以当前位置的输入 $\boldsymbol{x}_t$、之前位置的状态 $\boldsymbol{h}_{t-1}$ 为输入，以当前位置的状态 $\boldsymbol{h}_t$ 为输出的函数，按以下方式计算。

$$\boldsymbol{r}_t = \sigma\left(\boldsymbol{U}_r \cdot \boldsymbol{h}_{t-1} + \boldsymbol{W}_r \cdot \boldsymbol{x}_t + \boldsymbol{b}_r\right)$$

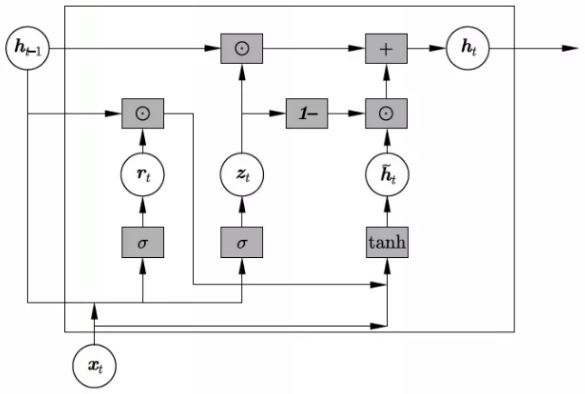
$$\boldsymbol{z}_t = \sigma\left(\boldsymbol{U}_z \cdot \boldsymbol{h}_{t-1} + \boldsymbol{W}_z \cdot \boldsymbol{x}_t + \boldsymbol{b}_z\right)$$

$$\tilde{\boldsymbol{h}}_t = \tanh\left(\boldsymbol{U}_h \cdot \boldsymbol{r}_t \odot \boldsymbol{h}_{t-1} + \boldsymbol{W}_h \cdot \boldsymbol{x}_t + \boldsymbol{b}_h\right)$$

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \tilde{\boldsymbol{h}}_t + \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1}$$

这里 $\boldsymbol{r}_t$ 是重置门，$\boldsymbol{z}_t$ 是更新门，$\tilde{\boldsymbol{h}}_t$ 是中间结果。状态 $\boldsymbol{h}_t$、重置门 $\boldsymbol{r}_t$、更新门 $\boldsymbol{z}_t$ 都是向量，其维度相同。

如下为 GRU 的单元示意图：



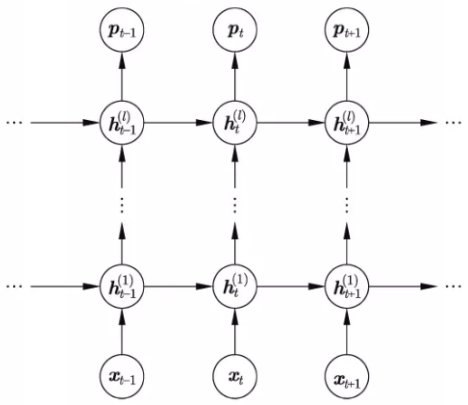
当 $\boldsymbol{z}_t = 0$，$\boldsymbol{r}_t = 1$ 时，GRU 回退到基本 RNN；

当 $\boldsymbol{z}_t = 0$，$\boldsymbol{r}_t = 0$ 时，$\boldsymbol{h}_t$ 只依赖于当前位置输入 $\boldsymbol{x}_t$，忽视之前位置的状态 $\boldsymbol{h}_{t-1}$；

当 $\boldsymbol{z}_t = 1$ 时，将之前位置的状态 $\boldsymbol{h}_{t-1}$ 复制到当前位置，忽视当前位置输入 $\boldsymbol{x}_t$。

3）深度循环神经网络

简单 RNN 只有一个隐层或者中间层，可以拓展到多个隐层的神经网络。多个隐层的状态之间存在层次化关系，模型具有更强的表示能力。

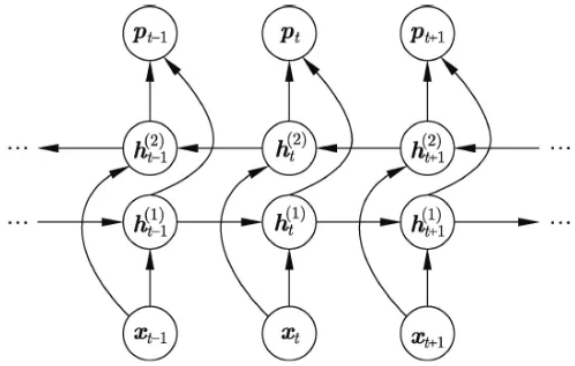


第 1 个隐层：$\boldsymbol{h}_t^{(1)} = \tanh\left(\boldsymbol{U}^{(1)} \cdot \boldsymbol{h}_{t-1}^{(1)} + \boldsymbol{W}^{(1)} \cdot \boldsymbol{x}_t + \boldsymbol{b}^{(1)}\right)$；

第 l 个隐层：$\boldsymbol{h}_t^{(l)} = \tanh\left(\boldsymbol{U}^{(l)} \cdot \boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{W}^{(l)} \cdot \boldsymbol{h}_t^{(l-1)} + \boldsymbol{b}^{(l)}\right)$；输出层为：$\boldsymbol{p}_t = \text{softmax}\left(\boldsymbol{V} \cdot \boldsymbol{h}_t^{(l)} + \boldsymbol{c}\right)$。

4）双向循环神经网络

简单 RNN 描述序列数据单方向的顺序依存关系，可以拓展到双方向。引入前向的循环神经网咯和后向的循环神经网路，在每一个位置将两个神经网络的状态向量拼接，构成新的状态向量。拼接的向量能结合两个方向的依存关系更好地表示序列数据的全局特征，模型具有更强的表示能力。



前向的隐状态为 $\boldsymbol{h}_t^{(1)} = \tanh\left(\boldsymbol{U}^{(1)} \cdot \boldsymbol{h}_{t-1}^{(1)} + \boldsymbol{W}^{(1)} \cdot \boldsymbol{x}_t + \boldsymbol{b}^{(1)}\right)$，

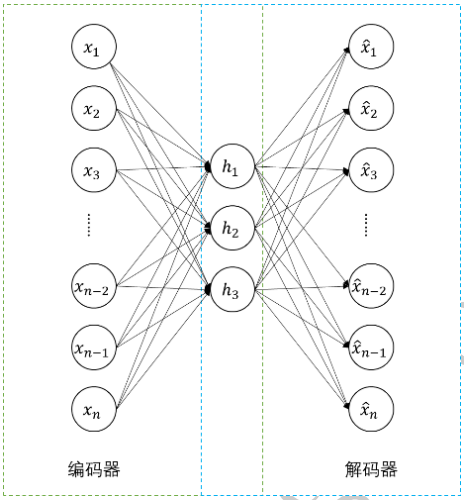
后向的隐状态为 $\boldsymbol{h}_t^{(2)} = \tanh\left(\boldsymbol{U}^{(2)} \cdot \boldsymbol{h}_{t+1}^{(2)} + \boldsymbol{W}^{(2)} \cdot \boldsymbol{x}_t + \boldsymbol{b}^{(2)}\right)$，两者拼接在一起的向量是 $\boldsymbol{h}_t = \left[\boldsymbol{h}_t^{(1)}; \boldsymbol{h}_t^{(2)}\right]$，输出为 $\boldsymbol{p}_t = \text{softmax}\left(\boldsymbol{V} \cdot \boldsymbol{h}_t + \boldsymbol{c}\right)$。

14 自编码器和生成模型

14.1 自编码器与限制玻尔兹曼机

14.1.1 自编码器

自编码器（Autoencoder）是一种无监督学习模型，其作用是将学习得到的数据进行有效表示（解码）。比如我们通过 PCA 将一个 10 维的数据降到 3 个维度，自编码器的作用就是把这 3 个维度再还原成我们原本的 10 个维度。如下图展示了从编码到解码的过程：



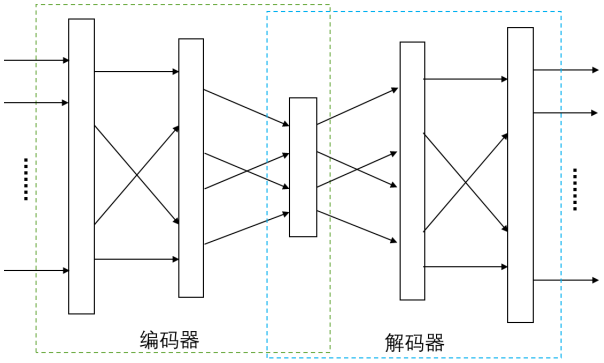
编码器负责编码到隐层，对样本进行低维表示：$\boldsymbol{h} = f(\boldsymbol{x})$，**解码器**部分：$g(\boldsymbol{h}) = \hat{\boldsymbol{x}}$。自编码器的目标是最小化输入数据 $\boldsymbol{x}$ 与重构数据 $\hat{\boldsymbol{x}}$ 之间的差异，通常使用均方误差（MSE）作为损失函数。

$$\min L\big(\boldsymbol{x}, g(f(\boldsymbol{x}))\big) = \frac{1}{N}\sum_{i=1}^{N} \|g(f(\boldsymbol{x}_i)) - \boldsymbol{x}_i\|^2$$

若采用**线性函数**，自编码器等价于主成分分析，隐层编码为前几个主成分；采用如 **Sigmoid 的非线性传递函数**时，可看作是用神经网络实现非线性主成分分析。自编码器的训练仍可采用**误差反向传播**算法。

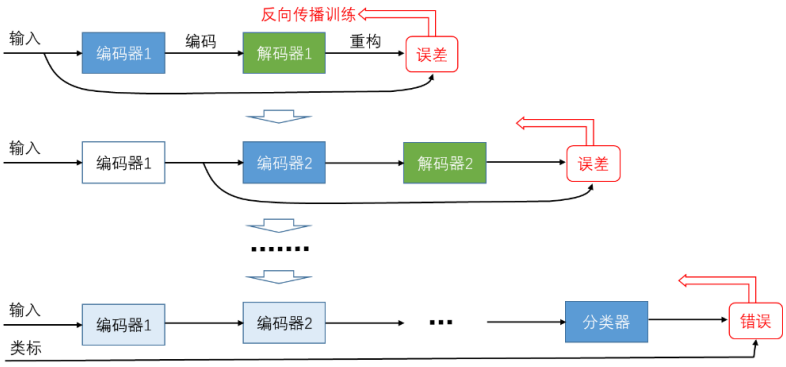
表示学习：通过**神经网络**学习得到对样本更有效的特征表示。

多层自编码器增加了多个隐层，用中间隐层（瓶颈层）节点输出作为对样本的低维表示（编码）。多层网络通过多层非线性叠加，可以拟合任意复杂的非线性映射。

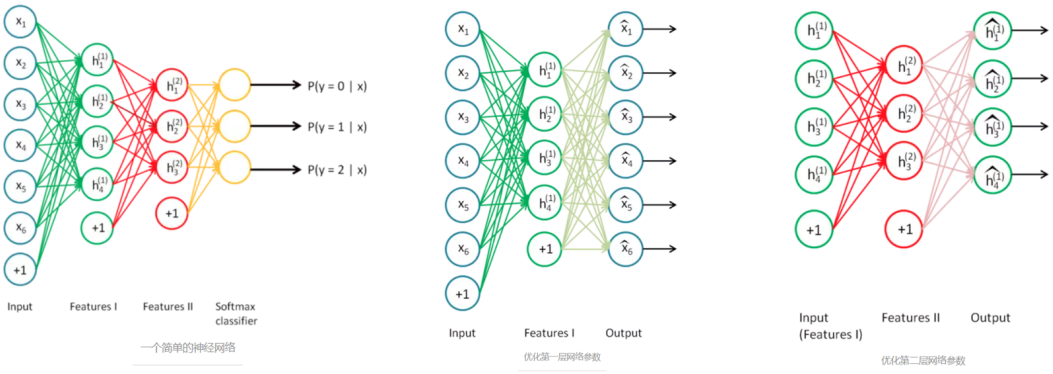


多层自编码器还可以用于构造深度学习网络。我们知道深度网络由于反向传播算法，很容易梯度消失或梯度爆炸，为了避免梯度消失的情况，也有结合自编码器一层一层对网络进行预训练的方法，称为**逐层贪婪训练法**。

- 先构造并训练一层自编码器；
- 去掉解码器层，再接一个新的简单编码器进行训练，训练完成后再把解码器层去掉；
- 以此类推，一步步增加新的编码器层，然后进行误差反向传播进行监督训练。



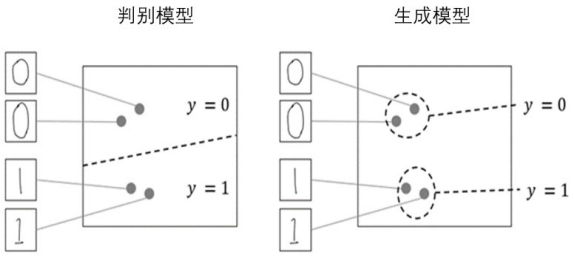
上述流程可用如下一个简单的神经网络理解，首先只保留输入层 Input 和第一个隐藏层 Features I 构成自编码器，训练这个自编码器得到第一层网络的参数（绿线部分）；接着第二层，只保留第一个隐藏层 Features I 和第二个隐藏层 Features II 再构成自编码器，训练这个自编码器得到第二层网络的参数（红线部分）。一层一层逐步优化后，我们把优化后的网络参数作为神经网络的初始值，之后微调（即训练过程）整个网络。



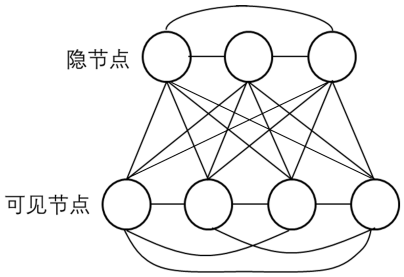
14.1.2 玻尔兹曼机

概率模型：基于概率论与统计学的框架，用于描述数据生成过程或对未知变量进行推断，建模观测数据 $\boldsymbol{x}$ 的概率分布 $p(\boldsymbol{x})$。可分为**判别模型**——对后验概率 $P(y|\boldsymbol{x})$ 直接建模；**生成模型**——估计或模拟样本的概率分布，对于分类问题，即计算联合概率分布 $p(\boldsymbol{x}, y)$。

简单点理解，**判别模型**就是找到分界把两类东西分开，比如我们熟知的 Logistic Regression、支持向量机（SVM）；而**生成模型**就是先学会每一类数据长啥样，然后根据学习可以完整地还原或生成样本，如朴素贝叶斯、隐马尔可夫模型、生成对抗网络（GAN）。下图直观展示了两者的作用：



玻尔兹曼机（BM）是一种受统计力学玻尔兹曼分布启发的多层神经网络，属于一种**生成模型**，能够生成新的数据样本。由**可见节点** $\boldsymbol{x}$ 和**隐节点** $\boldsymbol{h}$ 组成，两者都有**二值特征**（只有两种取值，0 或 1）。层与层之间、同层之间结点都是**全连接**。



玻尔兹曼分布是统计力学和概率论中一个重要的概念，用于描述系统中粒子或状态在不同能量水平上的分布情况。即一个系统在热平衡状态下，处于某个特定能量状态的概率。系统中状态 x 有能量 $E(x)$，该状态的概率 $p(x)$ 为：

$$p(x) = \frac{e^{-\frac{E(x)}{kT}}}{Z}$$

其中，k 为玻尔兹曼常数，T 为系统的绝对温度，Z 为归一化常数，确保概率总和为 1。

定义 Z 为：

$$Z = \sum_{x} e^{-\frac{E(x)}{kT}} \quad \text{（离散情况）}; \qquad Z = \int e^{-\frac{E(x)}{kT}} dx \quad \text{（连续情况）}$$

该分布描述了热力学粒子系统规律——粒子系统某一状态出现的概率，只由该状态的**物理能量**决定；**能量越低**，状态越稳定、**出现概率越高**。

借助上述的玻尔兹曼分布，玻尔兹曼机抛弃真实物理粒子，把图像 / 数据节点组合当成“系统状态”，人为自定义一个数学打分值，依旧命名为能量 E。

$E(\boldsymbol{x})$ 表示可见节点为 $\boldsymbol{x}$ 时的能量，概率 $p(\boldsymbol{x})$ 与能量的关系表示为

$$p(\boldsymbol{x}) = \frac{e^{-E(\boldsymbol{x})}}{Z}; \qquad \text{归一化因子} Z = \sum_{\boldsymbol{x}} e^{-E(\boldsymbol{x})}$$

能量 $E(\boldsymbol{x})$ 需考虑所有可能隐节点向量的取值（遍历隐节点 h 所有可能的 0/1 取值）：

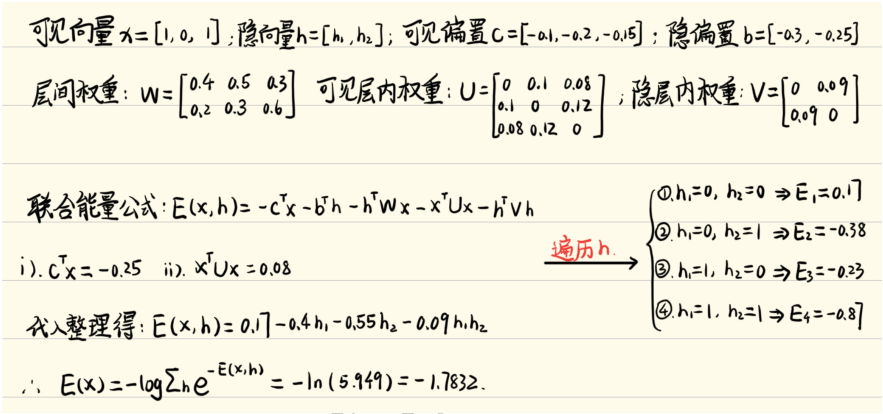
$$E(\boldsymbol{x}) = -\log \sum_{h} e^{-E(\boldsymbol{x},h)}$$

对可见节点向量 $\boldsymbol{x}$ 和隐节点向量 $\boldsymbol{h}$，能量定义为：

$$E(\boldsymbol{x},\boldsymbol{h}) = -\boldsymbol{c}^T\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{h} - \boldsymbol{h}^T W\boldsymbol{x} - \boldsymbol{x}^T U\boldsymbol{x} - \boldsymbol{h}^T V\boldsymbol{h}$$

$\boldsymbol{b}$ 和 $\boldsymbol{c}$ 分别为隐节点和可见节点的**偏置参数**；$\boldsymbol{W}$ 为可见节点和隐节点之间的连接权值矩阵；$\boldsymbol{U}$ 和 $\boldsymbol{V}$ 分别是可见节点和隐节点各自**内部的连接权值矩阵**

如下简单例子展示如何去计算一个可见节点 $\boldsymbol{x}$ 的能量：

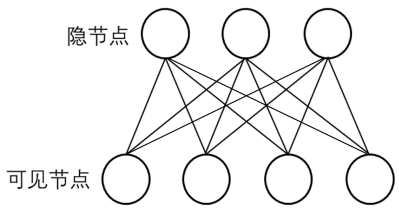


由于玻尔兹曼机存在三层连接，其同时考虑层间和层内的全连接目的是为了完全真实还原数据里所有变量的相互影响。但是该方法参数量很大，计算十分复杂，难以训练，因此考虑将层内的全连接去除。

写在前面：（限制性）玻尔兹曼机是用于发现数据中的隐藏特征，然后尽量去模仿真实数据的模型。比如我们的真实数据是一堆 0/1 组成的图片（像 MNIST 手写数字），我们希望模型也能生成类似的图片。因此，需要不断地调整模型的三个参数 $\boldsymbol{W}$、$\boldsymbol{b}$ 和 $\boldsymbol{c}$

14.1.3 限制性玻尔兹曼机

限制性玻尔兹曼机（RBM）结构与玻尔兹曼机相似，但是连接限制在可见层和隐层之间，**同层直接无连接**。



联合概率为

$$p(\boldsymbol{x},\boldsymbol{h})=\frac{e^{-E(\boldsymbol{x},\boldsymbol{h})}}{Z};\quad \text{归一化因子为}Z=\sum_{x,h}e^{-E(x,h)}$$

由于层内没有了连接，能量可简化为

$$E(\boldsymbol{x},\boldsymbol{h})=-\boldsymbol{c}^T\boldsymbol{x}-\boldsymbol{b}^T\boldsymbol{h}-\boldsymbol{h}^TW\boldsymbol{x}=-\sum_j\sum_k W_{j,k}h_jx_k-\sum_k c_kx_k-\sum_j b_jh_j$$

由此，可写成

$$p(\boldsymbol{x},\boldsymbol{h})=\frac{1}{Z}e^{(\sum_j\sum_k W_{j,k}h_jx_k+\sum_k c_kx_k+\sum_j b_jh_j)}=\frac{1}{Z}\prod_j\prod_k e^{W_{j,k}h_jx_k}\prod_k e^{c_kx_k}\prod_j e^{b_jh_j}$$

其中 h_j 和 x_k 分别表示隐层第 j 个节点和输入层第 k 个节点。

- **估计隐节点**：给定 $\boldsymbol{x}$ 下 $\boldsymbol{h}$ 的条件概率

$$p(\boldsymbol{h}|\boldsymbol{x})=\prod_j p(h_j|\boldsymbol{x})=\prod_j \text{sigm}\left((2h_j-1)(b_j+\boldsymbol{W}_{j\cdot}\boldsymbol{x})\right)$$

其中 sigm 为 sigmoid 函数，$\boldsymbol{W}_{j\cdot}$ 表示权值矩阵中第 j 个行向量。

推导如下：

$$\begin{aligned}
p(\boldsymbol{h}|\boldsymbol{x})&=\frac{p(\boldsymbol{x},\boldsymbol{h})}{p(\boldsymbol{x})}=\frac{p(\boldsymbol{x},\boldsymbol{h})}{\sum_h p(\boldsymbol{x},\boldsymbol{h})}=\frac{\frac{1}{Z}e^{-E(\boldsymbol{x},\boldsymbol{h})}}{\sum_h\frac{1}{Z}e^{-E(x,h)}}\\
&=\frac{e^{\sum_j\sum_k W_{j,k}h_jx_k+\sum_k c_kx_k+\sum_j b_jh_j}}{\sum_h e^{\sum_j\sum_k W_{j,k}h_jx_k+\sum_k c_kx_k+\sum_j b_jh_j}}=\frac{e^{\sum_k c_kx_k}e^{\sum_j W_{j\cdot}h_j\boldsymbol{x}+\sum_j b_jh_j}}{\sum_h e^{\sum_k c_kx_k}e^{\sum_j W_{j\cdot}h_j\boldsymbol{x}+\sum_j b_jh_j}}\\
&\xrightarrow{x\text{给定}}=\frac{e^{\sum_j W_{j\cdot}h_j\boldsymbol{x}+\sum_j b_jh_j}}{\sum_h e^{\sum_j W_{j\cdot}h_j\boldsymbol{x}+\sum_j b_jh_j}}=\frac{e^{\sum_j h_j(W_{j\cdot}\boldsymbol{x}+b_j)}}{\sum_h e^{\sum_j h_j(W_{j\cdot}\boldsymbol{x}+b_j)}}\\
&=\frac{\prod_j e^{h_j(W_{j\cdot}\boldsymbol{x}+b_j)}}{\sum_h\prod_j e^{h_j(W_{j\cdot}\boldsymbol{x}+b_j)}}=\frac{\prod_j e^{h_j(W_{j\cdot}\boldsymbol{x}+b_j)}}{\prod_j\sum_{h_j\in\{0,1\}} e^{h_j(W_{j\cdot}\boldsymbol{x}+b_j)}}\\
&=\prod_j\frac{e^{h_j(W_{j\cdot}\boldsymbol{x}+b_j)}}{\sum_{h_j\in\{0,1\}} e^{h_j(W_{j\cdot}\boldsymbol{x}+b_j)}}=\prod_j p(h_j|\boldsymbol{x})
\end{aligned}$$

$$\begin{cases}
\text{当}h_j=0, & \dfrac{e^{h_j(W_{j\cdot}\boldsymbol{x}+b_j)}}{\sum_{h_j\in\{0,1\}} e^{h_j(W_{j\cdot}\boldsymbol{x}+b_j)}}=\dfrac{e^0}{e^0+e^{W_{j\cdot}\boldsymbol{x}+b_j}}\\
\text{当}h_j=1, & \dfrac{e^{h_j(W_{j\cdot}\boldsymbol{x}+b_j)}}{\sum_{h_j\in\{0,1\}} e^{h_j(W_{j\cdot}\boldsymbol{x}+b_j)}}=\dfrac{e^{W_{j\cdot}\boldsymbol{x}+b_j}}{e^0+e^{W_{j\cdot}\boldsymbol{x}+b_j}}
\end{cases}$$

选取 $\text{sigm}(z)=\dfrac{1}{1+e^{-z}}$，当 $z=(2h_j-1)(W_{j\cdot}\boldsymbol{x}+b_j)$ 时满足两种情况的形式。

$\therefore\ p(\boldsymbol{h}|\boldsymbol{x})=\prod_j p(h_j|\boldsymbol{x})=\prod_j\text{sigm}\left((2h_j-1)(W_{j\cdot}\boldsymbol{x}+b_j)\right)$

- **生成观测数据**：给定 $\boldsymbol{h}$ 下 $\boldsymbol{x}$ 的条件概率

$$p(\boldsymbol{x}|\boldsymbol{h}) = \prod_k p(x_k|\boldsymbol{h}) = \prod_k \text{sigm}\left((2x_k - 1)(c_k + \boldsymbol{h}\boldsymbol{W}_{\cdot k})\right)$$

这里的推导过程和思路与上面估计隐节点部分高度相似：

因$\boldsymbol{h}$给定：

$$\begin{aligned}
p(\boldsymbol{x}, \boldsymbol{h}) &= \frac{1}{Z} e^{\left(\sum_j \sum_k W_{jk} h_j x_k + \sum_k c_k x_k + \sum_j b_j h_j\right)} = \frac{1}{Z} e^{\sum_j b_j h_j} \cdot e^{\sum_j \sum_k W_{jk} h_j x_k + \sum_k c_k x_k} \\
&= \frac{e^{\sum_j b_j h_j}}{Z} e^{\sum_k W_{\cdot k} \boldsymbol{h} x_k + \sum_k c_k x_k} = \frac{e^{\sum_j b_j h_j}}{Z} e^{\sum_k x_k (W_{\cdot k} \boldsymbol{h} + c_k)} \\
&= \frac{e^{\sum_j b_j h_j}}{Z} \prod_k e^{x_k (c_k + \boldsymbol{h}\boldsymbol{W}_{\cdot k})} \\
\therefore\ p(\boldsymbol{x}|\boldsymbol{h}) &= \frac{p(\boldsymbol{x}, \boldsymbol{h})}{p(\boldsymbol{h})} = \frac{p(\boldsymbol{x}, \boldsymbol{h})}{\sum_x p(\boldsymbol{x}, \boldsymbol{h})} = \frac{\prod_k e^{x_k (c_k + \boldsymbol{h}\boldsymbol{W}_{\cdot k})}}{\sum_x \prod_k e^{x_k (c_k + \boldsymbol{h}\boldsymbol{W}_{\cdot k})}} \\
&= \frac{\prod_k e^{x_k (c_k + \boldsymbol{h}\boldsymbol{W}_{\cdot k})}}{\prod_k \sum_{x_k \in \{0,1\}} e^{x_k (c_k + \boldsymbol{h}\boldsymbol{W}_{\cdot k})}} = \prod_k \frac{e^{x_k (c_k + \boldsymbol{h}\boldsymbol{W}_{\cdot k})}}{\sum_{x_k \in \{0,1\}} e^{x_k (c_k + \boldsymbol{h}\boldsymbol{W}_{\cdot k})}} \\
&= \prod_k p(x_k|\boldsymbol{h}) = \prod_k \text{sigm}\left((2x_k - 1)(c_k + \boldsymbol{h}\boldsymbol{W}_{\cdot k})\right)
\end{aligned}$$

接下来是如何用样本 $\boldsymbol{x}$ 对 RBM 进行**参数学习**：

目标函数：对所有可能隐节点状态求 $\boldsymbol{x}$ 的边缘分布

$$P(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{\boldsymbol{h}} p(\boldsymbol{x}, \boldsymbol{h}) = \sum_{\boldsymbol{h}} \frac{e^{-E(\boldsymbol{x}, \boldsymbol{h})}}{Z}$$

最大似然求解最优化参数，调整参数让输入的样本概率最大化

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{x}; \boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \big(-\log P(\boldsymbol{x})\big)$$

梯度下降求解：

$$\frac{\partial(-\log P(x))}{\partial \theta} = \sum_h p(h|x) \frac{\partial E(x,h)}{\partial \theta} - \sum_{x,h} p(x,h) \frac{\partial E(x,h)}{\partial \theta}$$

也可写为：

$$\frac{\partial(-\log P(x))}{\partial \theta} = \mathbb{E}_{P(h|x)} \frac{\partial E(x,h)}{\partial \theta} - \mathbb{E}_{P(x,h)} \frac{\partial E(x,h)}{\partial \theta}$$

推导过程如下：

$$\text{-log}\, P(x) = -\log\left(\sum_h \frac{e^{-E(x,h)}}{Z}\right) = -\log\left(\sum_h e^{-E(x,h)}\right) + \log Z$$

$$\frac{\partial(-\log P(x))}{\partial \theta} = \frac{\partial\left(-\log\left(\sum_h e^{-E(x,h)}\right)\right)}{\partial \theta} + \frac{\partial(\log Z)}{\partial \theta}$$

前部分：

$$\begin{aligned}\frac{\partial\left(-\log\left(\sum_h e^{-E(x,h)}\right)\right)}{\partial\theta} &= -\frac{1}{\sum_h e^{-E(x,h)}}\cdot\frac{\partial\left(\sum_h e^{-E(x,h)}\right)}{\partial\theta}\\ &= \frac{-\sum_h\left(e^{-E(x,h)}\frac{\partial E(x,h)}{\partial\theta}\right)}{\sum_h e^{-E(x,h)}} = \sum_h\left(\frac{e^{-E(x,h)}}{\sum_h e^{-E(x,h)}}\cdot\frac{\partial E(x,h)}{\partial\theta}\right)\\ &= \sum_h\left(\frac{\frac{e^{-E(x,h)}}{Z}}{\frac{\sum_h e^{-E(x,h)}}{Z}}\cdot\frac{\partial E(x,h)}{\partial\theta}\right) = \sum_h\left(\frac{p(x,h)}{p(x)}\cdot\frac{\partial E(x,h)}{\partial\theta}\right)\\ &= \sum_h p(h|x)\frac{\partial E(x,h)}{\partial\theta}.\end{aligned}$$

后部分：

$$\begin{aligned}\frac{\partial(\log Z)}{\partial\theta} &= \frac{\partial\left(\log\sum_{x,h}e^{-E(x,h)}\right)}{\partial\theta} = \frac{1}{\sum_{x,h}e^{-E(x,h)}}\cdot\frac{\partial\left(\sum_{x,h}e^{-E(x,h)}\right)}{\partial\theta}\\ &= \frac{1}{\sum_{x,h}e^{-E(x,h)}}\cdot\sum_{x,h}\left(-e^{-E(x,h)}\cdot\frac{\partial E(x,h)}{\partial\theta}\right)\\ &= \sum_{x,h}\left(\frac{-e^{-E(x,h)}}{\sum_{x,h}e^{-E(x,h)}}\cdot\frac{\partial E(x,h)}{\partial\theta}\right) = -\sum_{x,h}p(x,h)\cdot\frac{\partial E(x,h)}{\partial\theta}\end{aligned}$$

$$\therefore\ \frac{\partial(-\log P(x))}{\partial\theta} = \sum_h p(h|x)\frac{\partial E(x,h)}{\partial\theta} - \sum_{x,h}p(x,h)\cdot\frac{\partial E(x,h)}{\partial\theta}$$

$\mathbb{E}_{P(Y)}[f(Y)]$ 表示离散随机变量 Y 在分布 $P(Y)$ 下的数学期望。

$$\frac{\partial(-\log P(x))}{\partial\theta} = \mathbb{E}_{P(h|x)}\frac{\partial E(x,h)}{\partial\theta} - \mathbb{E}_{P(x,h)}\frac{\partial E(x,h)}{\partial\theta}$$

由此可以对各参数进行更新：

W:

$$\frac{\partial(-\log P(x))}{\partial W} = -\mathbb{E}_h\big(h(x(t))^T x(t)\big) + \mathbb{E}_{x,h}\big(h^T x\big)$$

c:

$$\frac{\partial(-\log P(x))}{\partial c} = -x(t) + \mathbb{E}_{x,h}\big(x\big)$$

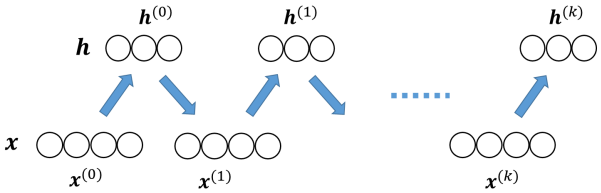
$\boldsymbol{b}$:

$$\frac{\partial(-\log P(\boldsymbol{x}))}{\partial \boldsymbol{b}} = -\mathbb{E}_{\boldsymbol{h}}\big(\boldsymbol{h}(\boldsymbol{x}(t))\big) + \mathbb{E}_{\boldsymbol{x},\boldsymbol{h}}\big(\boldsymbol{h}\big)$$

这里的期望值 $\mathbb{E}_{\boldsymbol{h}}() = \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{x})}()$ 可根据 $p(\boldsymbol{h}|\boldsymbol{x})$ 求得，但是 $\mathbb{E}_{\boldsymbol{x},\boldsymbol{h}}()$ 是对全部可能的 $\boldsymbol{x}$、$\boldsymbol{h}$ 联合分布求期望，难以精确计算。因此使用 **CD 学习算法** 来实现参数的更新。

- 用蒙特卡洛迭代的方式来对状态空间进行采样；
- k 步交替采样后得到的可见节点和隐节点状态 $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{h}})$ 上的值作为对期望 $\mathbb{E}_{\boldsymbol{x},\boldsymbol{h}}(\cdot)$ 的估计；

- 在 RBM 网络学习中，只需 k=1 即可得到比较理想的效果。



简单来说，CD 算法的流程就是先给定 $\boldsymbol{x}$，$p(\boldsymbol{h}|\boldsymbol{x})$ 可以一次性算出所有 $\boldsymbol{h}_j$ 的分布，采样出一组 $\boldsymbol{h}$；再在采样出的一组 $\boldsymbol{h}$ 的基础上，计算 $p(\boldsymbol{x}|\boldsymbol{h})$ 所有 $\boldsymbol{x}_k$ 的分布，再采样出一组 $\boldsymbol{x}$。重复这个交替采样的操作 k 轮，得到 $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{h}})$ 越来越贴近全局联合分布 $p(\boldsymbol{x}, \boldsymbol{h})$。流程如下：

- **初始化**：参数 $\boldsymbol{W}$、$\boldsymbol{h}$ 和 $\boldsymbol{c}$，设置学习率 η；
- 加入训练样本 $\boldsymbol{x}_0$，采样：

  a) 对所有隐节点，计算 $P(h_j = 1 \mid \boldsymbol{x}_0) = \text{sigm}(b_j + \boldsymbol{W}_{j\cdot}\boldsymbol{x}_0)$，依此采样得向量 $\boldsymbol{h}_0$

  b) 对所有可见节点，计算 $P(x_k = 1 \mid \boldsymbol{h}_0) = \text{sigm}(c_k + \boldsymbol{h}_0\boldsymbol{W}_{\cdot k})$，依此采样得向量 $\boldsymbol{x}_1$

  c) 对所有隐节点，计算 $P(h_j = 1 \mid \boldsymbol{x}_1) = \text{sigm}(b_j + \boldsymbol{W}_{j\cdot}\boldsymbol{x}_1)$，依此采样得向量 $\boldsymbol{h}_1$

  更新：

$$\boldsymbol{W} \leftarrow \boldsymbol{W} + \eta\big(\boldsymbol{h}_0^T\boldsymbol{x}_0 - \boldsymbol{h}_1^T\boldsymbol{x}_1\big)$$

$$\boldsymbol{c} \leftarrow \boldsymbol{c} + \eta\big(\boldsymbol{x}_0 - \boldsymbol{x}_1\big)$$

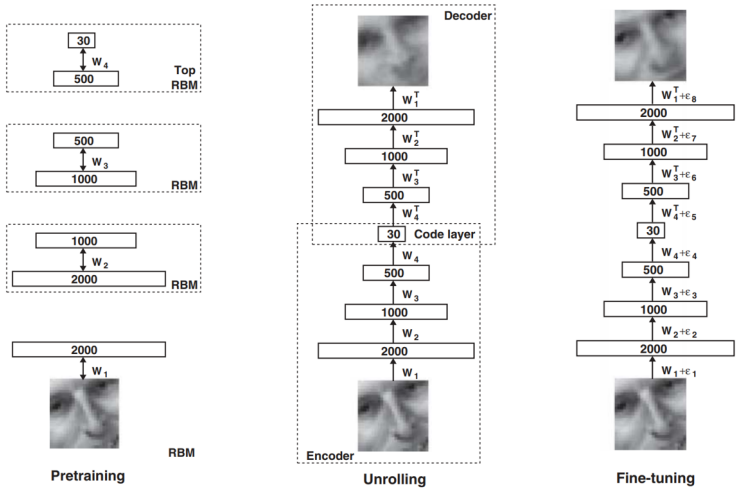
$$\boldsymbol{b} \leftarrow \boldsymbol{b} + \eta\big(\boldsymbol{h}_0 - \boldsymbol{h}_1\big)$$

- 检查收敛条件，如已收敛或已达预设训练次数则停止，否则重复第二步。

14.1.4 深度自编码器

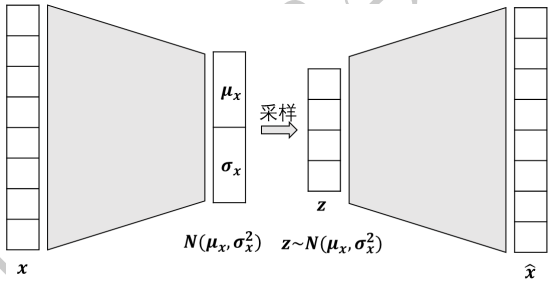
深度自编码器也采用了**逐层训练**的策略。深度自编码器达到非常有效的降维和关键特征提取效果。

- **预训练**：采用限制性玻尔兹曼机 RBM 对每一层编码层进行训练。先将输入节点和第一编码层节点看作 RBM 网络，CD 算法训练得到第一层的权值矩阵 $\boldsymbol{W}_1$；再把第一编码层看作可见节点，第二编码层看作隐节点，计算第一层的权值矩阵 $\boldsymbol{W}_2$，以此类推。
- **解码器**：完成编码器的逐层 RBM 预训练训练后，深度自编码器把得到的权值矩阵**对称到解码器的对应各层**，相当于用隐层节点逐层反推可见节点，最后得到与原输入样本**维数相同**的解码向量。这个过程被称为“Unrolling”。
- **微调**（fine-tuning）：完成对深度自编码器的 RBM 非监督预训练，把输入数据和输出数据作为整个网络的输入和输出，用误差反向传播算法对网络进行训练，同步更新所有层的全部权重。



14.2 生成模型——变分自编码器

回顾本章节一开始的自编码器，其编码器的输出是隐向量 $\boldsymbol{h}$，而**变分自编码器**（VAE）的编码器输出是 (μ, σ)，再根据高斯分布采样（重参数技巧）得到随机向量 $\boldsymbol{z}$。这是因为普通的自编码器不能随机生成全新样本，只能重建输入过的图片；而变分自编码器中引入了概率模型，能够随机采样得到向量 $\boldsymbol{z}$，可以直接生成从未见过的新图片。如下图所示，该生成模型对样本 $\boldsymbol{x}$，从一组隐含向量 $\boldsymbol{z}$ 中依概率产生。



对于 VAE，我们的目标是知道在给定的 $\boldsymbol{x}$ 下，其对应的隐特征 $\boldsymbol{z}$ 服从的分布，去计算 $\boldsymbol{z}$ 的后验概率密度 $p(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x})=\int_{\boldsymbol{z}} p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})d\boldsymbol{z}}$，这里的 $p(\boldsymbol{x})$ 是观测样本 $\boldsymbol{x}$ 的边缘密度，难以求解，因此我们考虑变分函数去近似后验概率。

变分推断：对函数的微小变化（称作变分）寻找函数的极值。考虑引入某个易计算的函数变分函数 $q(\boldsymbol{z})$，使其尽可能接近 $p(\boldsymbol{z}|\boldsymbol{x})$。

量化概率密度函数之间的相似度——**KL 散度**

设 x 是一个离散的随机变量，它的概率分布函数是 $P(x)$；同一随机变量，还存在另一个概率分布函数 $Q(x)$，$P(x)$ 和 $Q(x)$ 的差异定义为 KL 散度：

$$D_{KL}(Q \parallel P) = \sum_x Q(x) \log Q(x) - \sum_x Q(x) \log P(x) = \sum_x Q(x) \log \frac{Q(x)}{P(x)} = -\sum_x Q(x) \log \frac{P(x)}{Q(x)}$$

$D_{KL}(Q \parallel P)$ 表示用近似分布 Q 的概率做加权，只关心 Q 认为高概率的区域里，P 和 Q 差多少。

一般情况下，$D_{KL}(P \parallel Q) \neq D_{KL}(Q \parallel P)$，且有非负性 $D_{KL}(P \parallel Q) \geq 0$。

根据变分函数 $q(\boldsymbol{z})$ 逼近难以求解的后验概率密度 $p(\boldsymbol{z}|\boldsymbol{x})$ 可知：

$$\begin{aligned} D_{KL}\big(q(\boldsymbol{z}) \parallel p(\boldsymbol{z}|\boldsymbol{x})\big) &= \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} = \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})p(\boldsymbol{x})}{p(\boldsymbol{x},\boldsymbol{z})} d\boldsymbol{z} \\ &= \int_{\boldsymbol{z}} q(\boldsymbol{z}) \Big(\log q(\boldsymbol{z}) + \log p(\boldsymbol{x}) - \log p(\boldsymbol{x},\boldsymbol{z}) \Big) d\boldsymbol{z} \\ &= \log p(\boldsymbol{x}) + \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log q(\boldsymbol{z}) d\boldsymbol{z} - \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log p(\boldsymbol{x},\boldsymbol{z}) d\boldsymbol{z} \end{aligned}$$

可得：

$$\begin{aligned} \log p(\boldsymbol{x}) &= D_{KL}\big(q(\boldsymbol{z}) \parallel p(\boldsymbol{z}|\boldsymbol{x})\big) - \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log q(\boldsymbol{z}) d\boldsymbol{z} + \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log p(\boldsymbol{x},\boldsymbol{z}) d\boldsymbol{z} \\ &= D_{KL}\big(q(\boldsymbol{z}) \parallel p(\boldsymbol{z}|\boldsymbol{x})\big) + \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z})} d\boldsymbol{z} \end{aligned}$$

由 $D_{KL}\big(q(\boldsymbol{z}) \parallel p(\boldsymbol{z}|\boldsymbol{x})\big)$ 非负性可知：

$$\log p(\boldsymbol{x}) \geq \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z})} d\boldsymbol{z} \mathcal{L}(q) \triangleq \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z})} d\boldsymbol{z}$$

$\mathcal{L}(q)$ 为 $\log p(\boldsymbol{x})$ 变分下界，称作证据下界，简写为 ELBO，最大化 ELBO 即等价于最小化 KL 散度。分解 $\mathcal{L}(q)$ 中联合概率：

$$\mathcal{L}(q) = \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log p(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z} + \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\boldsymbol{z})}{q(\boldsymbol{z})} d\boldsymbol{z} = \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}\big[\log p(\boldsymbol{x}|\boldsymbol{z})\big] - D_{KL}\big(q(\boldsymbol{z}) \parallel p(\boldsymbol{z})\big)$$

第一项为似然函数，第二项为 KL 散度；上式中 $q(\boldsymbol{z})$ 可看做在观测样本下从给定函数族中求解的，因此可以看作 $\boldsymbol{x}$ 到 $\boldsymbol{z}$ 的条件密度 $q(\boldsymbol{z}|\boldsymbol{x})$

目标：最大化 ELBO

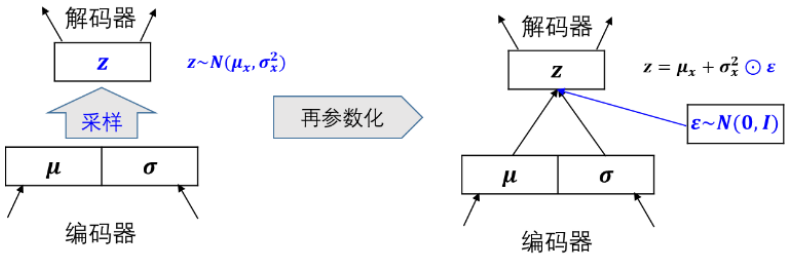
$$\max \mathcal{L}(q) = \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}\big[\log p(\boldsymbol{x}|\boldsymbol{z})\big] - D_{KL}\big(q(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\big)$$

指定隐层为各项独立的多元正态分布 $\mathcal{N}(\boldsymbol{\mu}, \sigma^2)$，则等价于：

$$\min \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2 + D_{KL}\big(q(\boldsymbol{z}|\boldsymbol{x}) \parallel \mathcal{N}(\boldsymbol{\mu}, \sigma^2)\big)$$

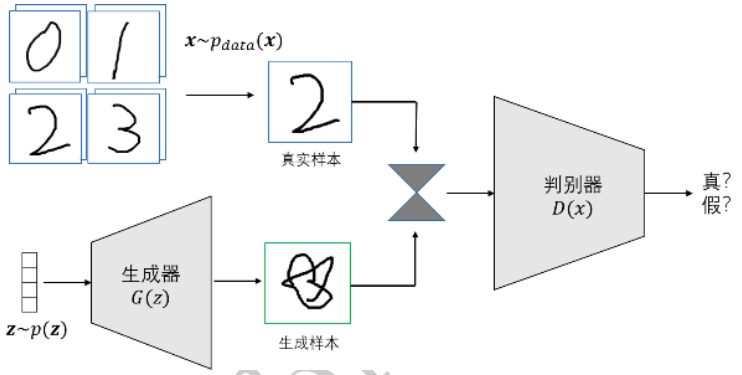
接下来便是完成对编码器和解码器的参数的更新，但由于网络中间存在采样，梯度无法反向传播到编码层，因此要解决这个问题，可选取**再参数化**的技巧。

- 原模型中解码层输入向量 $\boldsymbol{z}$ 是通过采样得到：$\boldsymbol{z}|\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x^2)$
- 再参数化，系数 ϵ 保留了随机性，但不是训练的参数，使得梯度可以传播：$\boldsymbol{z}|\boldsymbol{x} = \boldsymbol{\mu}_x + \boldsymbol{\sigma}_x \odot \boldsymbol{\varepsilon}$，　$\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$



14.3 生成模型——生成对抗网络

生成对抗网络（GAN）是一种基于博弈的生成模型，在图像生成等领域被广泛使用。其由生成网络和判别网络组成，生成网络自动生成数据，判别网络判断数据是**已给的**（真的）还是**生成的**（假的）。学习的目标是构建生成网络，能自动生成同已给训练数据同分布的数据；学习的过程就是生成网络和判别网络不断通过优化自己网络的参数进行博弈的过程。



- 生成器 $G(\boldsymbol{z})$：在一定的隐变量控制下生成新样本；
- 判别器 $D(\boldsymbol{x})$：对真实训练样本和生成器生成的“假样本”判别；
- $\boldsymbol{z}$ 的先验概率密度为 $p(\boldsymbol{z})$，$\boldsymbol{x}$ 的概率密度函数 $p_{data}(\boldsymbol{x})$。

学习目标：判别器要使目标函数最大化，生成器要使判别函数最小化

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\big[\log D(\boldsymbol{x})\big] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\Big[\log\big(1 - D(G(\boldsymbol{z}))\big)\Big]$$

GAN 分批随机梯度下降训练算法：

1. 对**判别器**进行 k（设置的超参数）步优化，对每一步：

   1）从生成器隐变量先验密度 $p_g(\boldsymbol{z})$ 中采样生成一批 m 个生成样本 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$；

   2）从训练样本集中采样一批 m 个真实样本 $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$；

   3）对判别器的参数 θ_D 求目标函数的梯度

$$\nabla_{\boldsymbol{\theta}_D} \frac{1}{m} \sum_{i=1}^{m} \Big(\log D(\boldsymbol{x}^{(i)}) + \log\big(1 - D(G(\boldsymbol{z}^{(i)}))\big) \Big)$$

用该梯度上升的方向更新判别器参数 θ_D 为

$$\boldsymbol{\theta}_D \leftarrow \boldsymbol{\theta}_D + \eta \nabla_{\boldsymbol{\theta}_D} \frac{1}{m} \sum_{i=1}^{m} \Big(\log D(\boldsymbol{x}^{(i)}) + \log \big(1 - D(G(\boldsymbol{z}^{(i)}))\big) \Big)$$

2. 对**生成器**进行优化：

1）从生成器的隐变量概率密度 $p_g(\boldsymbol{z})$ 中采样生成一批样本 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$；

2）对生成器参数 $\boldsymbol{\theta}_G$ 求目标函数的梯度

$$\nabla_{\boldsymbol{\theta}_G} \frac{1}{m} \sum_{i=1}^{m} \log \big(1 - D(G(\boldsymbol{z}^{(i)}))\big)$$

用该梯度下降的方向更新生成器参数 $\boldsymbol{\theta}_G$，即

$$\boldsymbol{\theta}_G \leftarrow \boldsymbol{\theta}_G - \eta \nabla_{\boldsymbol{\theta}_G} \frac{1}{m} \sum_{i=1}^{m} \log \big(1 - D(G(\boldsymbol{z}^{(i)}))\big)$$

3. 如此往复迭代训练，直到达到预设训练次数。

15 Transformer 模型简介

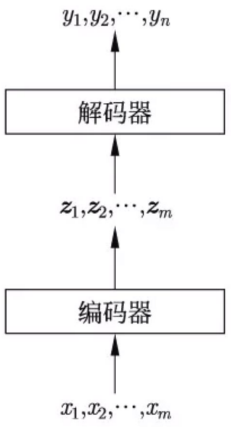
15.1 序列到序列的基本 RNN 模型

序列到序列学习是一套输入一段序列、输出另一段不等长序列的神经网络框架，如机器翻译、摘要生成、对话生成、语音识别等。这类任务是依赖**序列到序列模型**执行的，由**编码器**（将输入的单词序列转换成中间表示的序列）和**解码器**（将中间表示的序列转换成输出的单词序列）组成。

假设输入的单词序列是 $x_1, x_2, \ldots, x_m$，输出的单词序列是 $y_1, y_2, \ldots, y_n$；给定输入单词序列条件下输出单词序列的条件概率是

$$P(y_1, y_2, \ldots, y_n \mid x_1, x_2, \ldots, x_m) = \prod_{i=1}^{n} P(y_i \mid y_1, y_2, \ldots, y_{i-1}, x_1, x_2, \ldots, x_m)$$

这里 $P(y_1, y_2, \ldots, y_n \mid x_1, x_2, \ldots, x_m)$ 预测**给定输入序列**及**已生成输出序列**的条件下，**下一个位置上单词**出现的条件概率。如下图，序列到序列模型由编码器网络和解码器网络组成。



编码器将输入单词序列 $x_1, x_2, \ldots, x_m$ 转换成中间表示序列 $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m$，即

$$(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m) = F(x_1, x_2, \ldots, x_m)$$

解码器根据 $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m$ 依次生成输出单词序列 $y_1, y_2, \ldots, y_n$，即

$$P(y_i \mid y_1, y_2, \ldots, y_{i-1}, x_1, x_2, \ldots, x_m) = G(y_1, y_2, \ldots, y_{i-1}, \boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m)$$

这里的编码器和解码器都是我们先前所学习过的 RNN 模型，编码器将**最终位置的状态**表示传递到解码器，解码器根据得到的**编码器状态**决定其状态的序列以及输出的单词序列，RNN 通常选取 LSTM 和 GRU 能够更好地刻画长距离依存关系。

- **编码器（LSTM）**：

$$\boldsymbol{h}_j = a(\boldsymbol{x}_j, \boldsymbol{h}_{j-1}), \quad j = 1, 2, \ldots, m$$

$\boldsymbol{h}_j$ 是当前位置的状态；$\boldsymbol{h}_{j-1}$ 是前一个位置的状态；$\boldsymbol{x}_j$ 是当前位置的输入单词的词向量；a 是

处理单元，如 LSTM；$\boldsymbol{h}_0 = \mathbf{0}$

- **解码器（LSTM）**：

$$\boldsymbol{s}_i = a(\boldsymbol{y}_{i-1}, \boldsymbol{s}_{i-1}), \quad i = 1, 2, \ldots, n$$

$\boldsymbol{s}_i$ 是当前位置的状态；$\boldsymbol{s}_{i-1}$ 是前一个位置的状态；$\boldsymbol{y}_{i-1}$ 是前一个位置的输出词向量。

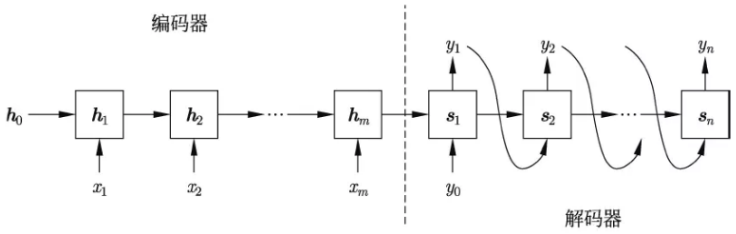
编码器将其最终状态 $\boldsymbol{h}_m$ 作为整个输入单词序列的表示传递给解码器。解码器将 $\boldsymbol{h}_m$ 作为解码器的初始状态 $\boldsymbol{s}_0$ 决定其状态序列。

- **输出（MLP）**：

$$\boldsymbol{p}_i = g(\boldsymbol{s}_i), \quad i = 1, 2, \ldots, n$$

$\boldsymbol{p}_i$ 是当前位置的输出；g 是输出层函数，由线性变换和软最大化函数组成；$\boldsymbol{p}_i$ 表示下一个位置单词出现的条件概率

如下图展示了序列到序列的基本 RNN 模型：



15.2 注意力机制

先前的基本 RNN 模型处理序列数据的时候，仅选用一**个中间表示**描述整个输入序列，其表示能力有限。要解决这个问题，就是利用注意力机制在**输出序列的每一个位置上产生一个组合输入序列的中间表示**。